University of California
Santa Cruz

Group Coordination Support in Networked Multimedia Systems

A dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy
in
Computer Engineering
by
Hans-Peter Dommel
December 1999

The dissertation of Hans-Peter Dommel is approved:

Prof. J. J. Garcia-Luna-Aceves

Prof. Patrick Mantey

Prof. Glen Langdon

Dean of Graduate Studies and Research

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
DEC 1999		**00-12-1999 to 00-12-1999**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Group Coordination Support in Networked Multimedia Systems	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **University of California at Santa Cruz,Department of Computer Engineering,Santa Cruz,CA,95064**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
The original document contains color images.

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**		**179**	

Contents

List of Figures

List of Tables

Group Coordination Support
in Networked Multimedia Systems

Hans-Peter Dommel

Abstract

Advances in computer hardware and networking technology have incited the deployment of large-scale group-oriented applications for delivery or interactive development of multimedia content in the Internet. There is a growing number of protocols and techniques for group communication and membership services in the IP-multicast framework, however, group coordination support for telecollaborative tasks such as videoconferencing or distributed interactive simulation has received little attention. In this dissertation, we address network control and coordination functions to orchestrate synchronous multimedia groupwork, establishing a sharing discipline on multimedia resources and guaranteeing consistency of distributed activities with ordered multicasting.

We introduce a formal framework for group coordination and a turn-taking abstraction useful for evaluating coordination protocols. Elemental design choices for group coordination architectures and the concept of aggregated processing of coordination information are discussed.

A floor control methodology is presented to implement concurrency control for interactive, rather than transactive cooperation among users. Floors are dynamically generated, ephemeral permissions for using discrete or continuous media such as GUI objects, audio and video channels, or remote instruments. Floor control regulates user interaction and bandwidth consumption by throttling sources in sending information flows according to receiver interest. A novel taxonomy of floor control protocols and a comparative throughput analysis show that large-scale group coordination is most effectively supported by hierarchical host organization. Two new protocols are presented to provide floor control in fully-connected

networks and in multicast trees, where the addition of group-relative address information permits more sophisticated coordination among end-nodes.

Finally, we discuss the problem of out-of-sequence delivery in multicasting. A new taxonomy for ordered multicasting protocols and a comparison of message complexities elicit the benefits of aggregated, ordered multicasting. We present a novel multicast ordering protocol for more efficient total-order delivery of messages from multiple sources to multiple, potentially overlapping receiver groups in multicast trees. Previous solutions require the computation of a separate propagation graph to structure ordering relations and incur high cost, if sources change frequently. Our proposed mechanism is more flexible and efficient, because it relays ordering information in accordance with the hierarchical organization of end-hosts maintained by an underlying tree-based reliable multicast protocol.

Acknowledgments

I wish to thank my advisor and sage, Professor J.J. Garcia-Luna-Aceves, with all my heart for believing in me and for the years of positive-spirited guidance, patience, and humor. It was a great honor and joy to be his student, and to learn and grow under his wings. His doors have always been open and his discipline for research and gentle leadership are a true inspiration. I cannot imagine a better "doctoral father" and role model to be guided by in academic life and I look forward to future collaboration. Heartfelt appreciation goes also to JJ's family for the warm welcoming and caring.

I am grateful to Professors Patrick Mantey and Glen Langdon for serving on my thesis committee and for their interest and time. In particular, I appreciate Pat Mantey's detailed review and insightful critique of the thesis and I look forward to future joint research projects. Thanks to the "Coco Group" for the years of togetherness, and to the faculty and staff at the UCSC Computer Science and Engineering department for their help and sharing. In particular, I thank Lynne Sheehan for helping me stay, and Professor Manfred Warmuth for his "focus!" encouragements. Finally, I wish to express my gratitude to all my teachers along the way for their legacy and gift of inspiration.

The text of this dissertation includes material that has been previously published [51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61]. I have received the Best Student Paper Award from the IEEE Systems, Man and Cybernetics Society at SMC'98 for paper [57]. The co-author listed in these publications, Professor J. J. Garcia-Luna-Aceves, has directed and supervised the research which forms the basis for the dissertation.

The financial and computing facility support from UCSC, ONR grant N-00014-92-J-1807, DARPA grants F19628-96-C-0038 and DAAB07-97-C-D607, and in particular from the Fulbright Program for the initial scholarship is gratefully acknowledged.

Personal Acknowledgments

I will always be grateful for the encouragement and true friendship that Velma Velázquez has provided me with during graduate school, in good and in challenging times. Roya Nassiri-Rahimi, Priya Ganguli, Paul Scheidt, Peter Cornelius, Manfred Warmuth, Barbara Warsavage, Barbara McCain, Jade Rice, Sheraz Omarazai, Jody Ho, Gina Engracia, Glenda Benevides, Jodi Rieger, Alexandre Brandweijn, Jane Mio, Lisa Swehla, Alpha Schramm, Qin Shen, the Hostetter family, Annette Shaff, Vanessa Fontana, Linde Martin, and Habib Krit have been dear friends and supporters in different times on the way. I am very thankful to Senseis Linda Holiday, Glen Kimoto, Jamie Zimron, Martha Jordan, Dov Nadel, and the community of North Bay Aikido, fostering my body, mind, and spirit with Ki and energizing "keiko". I also want to express my gratitude to my friends from the "Quatschtafel", the Board of Directors and staff of the Santa Cruz Hostel Society, and to my housemates, in particular Andrea Mächler and Kathleen McDill, for their caring and gift of a home away from home. Tom Klemzak, Troy and Jasun Tipton have been inspiring friends in music.

Many thanks also to my faithful family and friends far away, in particular to my big brother Klaus, my wonderful cousin Isa Steffen in Switzerland, my aunt Renate Stemmer, Christina Maass, Daniela Guicking, Lubi Zmijanek, Georgina and Koni Sonnabend, Lydia Grabowiecki, Doris Haslbeck, Bernd Scharrer, Christine Werber, Regine Drake-Nold, and Corinna Thiemann in Germany, and Riaz Ezmailzadeh and Masami Ueda in Japan.

A sincere thanks to all you important people in my life, mentioned and unmentioned, for your open hearts, spirits, and teachings, and the moments of sharing and growth that brought us together. You know who you are!

I have fond memories of my father Josef Dommel and my uncle Hans Stemmer for letting me be and follow my aspirations. Finally, and most importantly, I wish to thank my mother for all the years of unconditional love, patience, and support, especially after I set off for the New World in pursuit of my own interests. I dedicate this work to her.

Santa Cruz, December 1999,

Hans-Peter Dommel

NetWorking Man

I get up at seven, yeah
And I go to work at nine
I got no time for livin'
'Caus I'm networkin' all the time
It gives me wings and makes me feel nice
Trees and sparrows in front of my eyes

It seems to me I could life my creed
A lot better if I reserve what I need
I guess that's why they call me
They call me the networking man
I don't wear just one, but multi-casts
I know I'll find home on many paths

Bridges and links they surround me for good
I hook up near and far, free-spirit hub
Ton's of new ideas, never get tired
I'm loosely coupled, but always stay wired
Work in the net, that's what I like
Everythin's smooth, never a spike

They call me the networking man
I guess that's what I am
Secure is my home, Sun shines inside
I'm routin'n switchin' all the time
I'm active and linked, meshed like a star
No time to camp out, distance too far

Core of my life, always on quest
Virtual rendezvous, no time for rest
My digital life, it's acknowledging me
Through open gateways my spirit roams free
Syncing and caching, I'm on my way
A spread spectrum of wherever I stay

I get home diffused at ten o'clock
Network's down, evenin' shock
Like a saint I turn on the switch
Wireless box, guaranteed glitch
Caught in a web, always seem to be wonderin'
How to connect and not be asunder...

Well they call me the networking man

I guess that's what I am

(free after "Working Man" by Rush)

Chapter 1
Introduction

1.1 Motivation

The proliferation of Internet services in recent years, in particular the World Wide Web, indicates the high demand for sharing of information through computer networks. A wider distribution of the work force, in form of telecommuting and ubiquitous computing [231], the advent of networked multimedia, and less expensive technology have shifted *telecollaboration* into the spotlight of mainstream computing. Telecollaboration comes in many faces, such as email, instant messaging, chatting, simultaneous but independent work on shared documents, and real-time interaction on the same media or resources, qualified by the increasing degree of mutual awareness and the ability for instant information exchange and manipulation. Synchronous telecollaboration enables people in different geographic locations to share and jointly manipulate multimedia information in real-time and at various levels of granularity, bridging time and space. This aspect stands in contrast to legacy client-server applications such as Internet radio broadcast or video-on-demand, and to asynchronous, document-centric collaboration tools like email, instant messaging, or chat rooms. In this thesis, we focus on network support for synchronous multimedia groupwork for the Internet and large user groups. Cerf *et al.* [34] pointed out the importance of transatlantic collaboration infrastructures in a memorandum in 1991.

Shared resources in the telecollaborative workspace are, for example, video and audio channels, remotely controlled devices such as cameras or surgical instruments, objects in the

graphical user interface, or components of a document. Application areas are, for example, conferencing [197], collaborative virtual environments [40], distance learning [152], remote visualization [169], distributed interactive simulations (DIS) [70], collaboratories [127], distributed real-time gaming environments [49], and telemedicine [125, 171]. Telecollaboration faces two paradoxes. First, it is to better individual and joint productivity among users, and to provide innovative ways for remote communication; however, multimedia-enabled groupwork tools are often clumsily integrated and difficult to use. Second, while users aspire cooperation, they may knowingly or unknowingly contend for shared resources in synchronous work, due to technological or sociological reasons [126].

In contrast to stand-alone applications, where the user interacts only with a computer system, engineering of telecollaborative systems is much more complex because it involves user, network, and host-related issues, such as human factors, Quality-of-Service, and heterogeneous platforms for applications. The end-to-end interaction manifests itself between users, not end hosts, and users expect ideally a telecollaboration environment providing a quality of interaction close to a face-to-face meeting. Limitations in the availability and accessibility of resources in the shared workspace of a telecollaborative system create contention, competition, and conflict among users and make it necessary to deploy coordination mechanisms to reach consensus on how to jointly and effectively use the resources. Conflicts stalling the workflow may occur before and during resource allocation to users, as well as during actual usage. Telecollaborative services build on the provision of group coordination mechanisms. These manage access, manipulation, distribution and presentation issues between users and shared resources. Such coordination mechanisms are necessary to allow users to achieve individual goals in the context of group-centered remote interaction, when *telepresence* [22] substitutes for physical presence.

Software to support collaborative work, generally termed *groupware* [67], or workgroup computing software, referred initially only to systems supporting the asynchronous exchange of text-documents, but more recent connotations include multimedia-based, synchronous interaction. Groupware development is linked to studies on computer-mediated communi-

cation [160], and the discipline that motivates and validates groupware design, computer-supported cooperative work (CSCW) [183]. Tool design for CSCW must meet quantitative and qualitative constraints both from an engineering and human factors perspective. Figure 1.1 shows a four-sector classification of different groupware paradigms known to date, according to the level of interactivity and synchrony provided. Interactivity indicates the degree of sharing of resources, unilateral or bilateral, in a groupware paradigm, and synchrony indicates whether resources are shared at the same or different times and whether instant response is expected between participants. Our classification stands in contrast to prior typologies discerning between synchrony and explicit or implicit interaction [183], or between time, place and predictability [95].

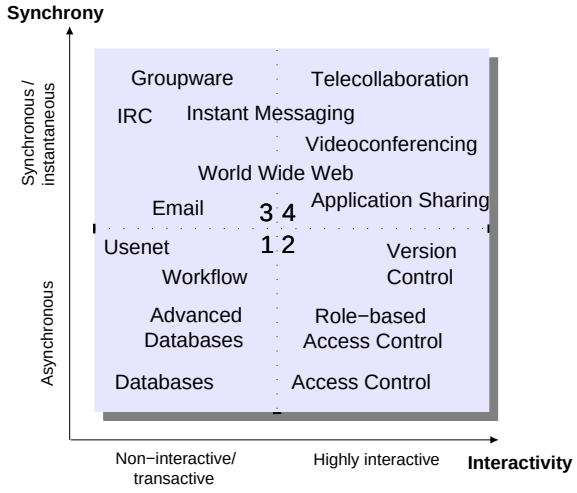

Figure 1.1: Classification of collaborative activities according to synchrony vs. interactivity.

Sector 1 targets non-interactive or transactive applications, whose task execution is asynchronous. This class is represented by database applications, which allow for multiparty access to shared records in a request-response scheme regulated by concurrency control. The objective of concurrency control [24] is to prevent conflicts on simultaneous read-write and write-write operations from several points of access to a shared database record. Users

have no insight into the actual information structure of database records and can only witness the outcome of transactions. While advanced database applications [20] support long-term interactivity on the same records, they also do not support end-user interactivity. Workflow systems [98], placed on the boundary between Sectors 1 and 3, implement typically asynchronous coordination and sharing mechanisms on task components belonging to a process flow among workflow participants. The goal is to administer task execution patterns in larger organizations, which can be predefined or generated ad hoc. Usenet access is database-driven and belongs to this sector, as well.

Sector 2, depicting synchronous, noninteractive systems, is represented by file-oriented control mechanisms such as access control [68, 202], role-based access control [193], or version control [123]. The shared resources are typically limited to discrete records, networked in file systems of local scope. Access is implemented in permission matrices based on a simple role distinction between the file owner, the group the owner belongs to, and all other users in the system. Access control lists [130] can additionally refine the security in concurrent file access, however, these mechanisms do not cater to frequently switching sources or continuous media streams, as we encounter them in networked multimedia systems.

Sector 3 depicts the predominant paradigms of current workgroup computing, including email, World Wide Web (WWW) access (via the Hypertext Transfer Protocol, or serving as a wrapper for other transport protocols), chat tools such as the Internet Relay Chat (IRC) [164] and their game-oriented extensions into "Multi-User Dungeons" (MUDs) or "Object-Oriented MUDs" (MOOs), and traditional groupware such as Lotus NOTES or Novell GROUPWISE. Multimedia support has been added to such systems in recent years, enabling users to interact via voice snippets, images, or short movies in specific digital video formats. User interaction is typically a request-reply pattern mediated by a client-server architecture. A more recent paradigm of Internet group communication is instant messaging (IM), which promotes highly interactive, lightweight exchange of information based on a presence protocol [194]. Users can detect at login time which other "friends" are online, and exchange small multimedia documents with near-instant notification and a

functionality similar to chat tools. Both WWW and IM approach Sector 4, however, the majority of tools are a far cry from synchronous multimedia collaboration functionality.

This dissertation concentrates on Sector 4, comprising tools for synchronous and interactive telecollaboration, videoconferencing, and application sharing. We target network support for real-time multiparty and multimedia interaction in packet-switched networks. Our intention is to improve on the state-of-the-art in network-centric and multimedia-oriented group collaboration. Within this category, we can distinguish between computer-supported realworld meeting spaces, where users are collocated and exchange information aided by computers, vs. virtual collaboration spaces, where users communicate and share information among geographically distributed workstations. We concentrate on the second paradigm, the distributed computer-mediated group collaboration with multimedia support. However, some of the methods put forward in this thesis are also applicable in computer-supported meeting spaces. Our working context is networked multimedia systems, consisting of multimedia-enabled hosts in a computer network exchanging media streams of various types.

1.2 Focus and Problem Formulation

The focus of this thesis is novel mechanisms for network-supported *group coordination*. Group coordination services support distributed hosts in coordinating their joint activities, to prevent or resolve resource contention, conflict and inconsistencies in the synchronous sharing of resources. Group coordination can be dissected in separate, yet related areas: the distributed access to shared resources in a networked multimedia system, which we also refer to as *floor control* [54], the ordered and reliable message dissemination [61], security in collaborative information exchange [88], and multisite synchronization [139] of distributed interaction. Group coordination protocols, which embrace multicasting and consider network conditions in the coordination processes between hosts, complement efforts on group membership known from distributed systems and multicasting as an efficient message dis-

semination mechanism for group communication. Such protocols may be implemented by man or machine, and in social or computer networking contexts.

We define coordination as an interactive scheduling process between two or more users forming a group to achieve joint work goals. Coordination correlates with cooperation, which we understand as the joint acting of individuals for a mutual benefit - in our context the mutual sharing of information for data mining or other forms of data exchange. Coordination and cooperation among users in networked multimedia systems support the process of *multimedia collaboration* [181], which is the actual act of users working together online. A system facilitating remote group collaboration with multimedia support is referred to as a *Collaborative Multimedia Environment* (CME). In contrast to earlier text-centered groupware systems, CMEs process multimodal resources, that is discrete and continuous data such as text, voice, video, music, sensor data from remote instruments, images and graphics.

A group is understood as an association of users with a common interest and for a common benefit. The timeframe within which a group meets in a distributed fashion is called a session, defining the modalities of information exchange between users. For example, a lecture session is comprised of a teacher and an audience consisting of students, where the lecture content is transmitted with a shared whiteboard, voice or video, and the teacher is primarily exercising control over these media. We take a process-oriented view on group coordination, meaning the design and analysis of distributed algorithms and protocols facilitating network-supported synchronous interaction, rather than looking at content and sharing functionality of specific coordination tools. Scalability of group coordination mechanisms is relevant in the light of observations that various MBone sessions [76] or DIS [70] can involve thousands of users and shared entities.

Such large-scale collaboration with multiple parties and mixed media over long distances raises complex coordination and cooperation issues. Currently, no comprehensive framework exists for characterizing the process of synchronous multimedia-based cooperation. Consequently, we experience a lack of methodology in the development and deployment of

coordination protocols for CMEs. A central point in understanding and successfully implementing CMEs is *telepresence* and support for mutual *awareness* [62, 133] among users. The challenge lies in the integration of usability issues concerning individual vs. group interests, heterogeneous operating systems and applications at end-hosts, and networking issues such as efficient session management and data dissemination.

From a user's perspective, computer-mediated remote interaction has several drawbacks in comparison to face-to-face meetings, because nonverbal communication across CME cannot be conveyed and perceived with the same subtlety as in face-to-face meetings due to limited image resolution, transmission delays, speaker identification and engagement problems. Gaze, as an important part in face-to-face communication, is estimated to be involved in over 60% of communication [147], serving as flow regulation, medium for communicational and emotional feedback, and reflecting the status of relationships between conversation participants. Gestures reportedly account for 35% of all interactions in order to enact ideas, signal turn-taking, or reference to objects [106]. Translating such subtle visual cues in face-to-face meetings onto the desktop metaphor, using network links as communication "bottlenecks", requires hence substitutive tools, for example cursors or icons to represent gestures, signals, and deixis. Studies on user interaction [114, 163] with a proprietary video conferencing system elicited some of the limits of video as a communication medium. Another key constraint in the effectiveness of groupwork with window-based human-computer interfaces is the limited screen space, despite improved screen and window management technology. Information from remote workspaces may be stripped from its source context and condensed into the shared workspace, causing "window thrashing" [120]. Users may experience cognitive overload, or become overinvolved in coordinating activities with each other to avoid conflict and hence get sidetracked from task completion. Consequently, telepresence becomes a poor substitute for physical presence, and limits the mutual awareness between users. Collaboration partners may disengage from groupwork, due to feelings of depersonalization, frustration or lack of trust, causing a halt or breakdown of collaboration efforts. Lack of coordination mechanisms compensating for such deficiencies, aside from

earlier technological limitations, may be one of the causes why collaborative applications have not been deployed successfully for large-scale group work among many users. To date it is an open problem, how online collaboration can be improved with such mechanisms, harnessing structured communication and compensating for lack of personal presence in synchronous groupwork, to foster synergy rather than obstructing the natural flow of user interaction [161].

From a distributed systems perspective, transaction processing [89] between users and a database server is based on the Commitment, Concurrency, and Recovery (CCR) infrastructure defined by ISO/IEC 10026, entailing atomicity, consistency, isolation, and durability (ACID) properties. Atomicity means that, to an outside observer, either all of the operations are completed or none of them is executed. Consistency warrants that the operations are performed correctly with respect to the application semantics. The isolation property means that any partial results of the operations composing the atomic action are not accessible before the completion of the atomic action. Finally, durability means that a transaction must endure a communication or an application failure. We observe that currently no such standard exists for building and deploying interaction management protocols for computer-supported cooperative work, independent of such middleware architectures as CORBA [153]. A group coordination architecture may provide a more lightweight repository for crafting applications to feature collaborative services transparently.

From a networking perspective, much research has been invested in recent years in designing more effective group communication mechanisms; however, without much consideration of end-user interactivity. While multicast [47] is a necessity for effective group communication (designed to have a source send a packet only once to the network interface, and multicast routers replicating the packet on its transmission path to multiple receivers), only few collaborative tools build on such efficient group communication to date. In addition, with IP multicast, no guarantees are given for reliable or order-preserving delivery of packets, and a message is delivered on a best-effort basis to all members of a multicast group. These shortcomings have spurred much research on multicast routing, reliable multicast,

Quality-of-Service, streaming protocols, or resource reservation. While group membership is tackled by such protocols as IGMP (Internet Group Management) [74] or session directory services such as SDP [102], and multipoint dissemination is handled by multicasting routing and reliable multicast protocols [135] to achieve efficient, reliable packet dissemination, group coordination support for interactive hosts has received only scattered attention in the research literature to date. We surmise that a formal and methodological framework for group coordination, providing an integrated view on dissemination, joint access and presentation issues from both a user and network perspective, is still lacking.

1.3 Objectives and Contributions

Our objective is to define a framework and architecture for group coordination, focusing on floor control and ordered multicast. The rationale behind the work presented in this thesis is that many networked multimedia applications exhibit similar media semantics and coordination needs, justifying the deployment of such services as middleware at a sub-application layer. Our goal is to achieve a better understanding of the group coordination problem, as a component of Internet multimedia collaboration, bridging the gap between CSCW and networking research. We envision a new generation of CME based on improved group coordination protocols, facilitating multipoint, multiparty, multichannel, and multimedia communication from small to very large groups up to an Internet scope. In such systems, groups and individuals can selectively, securely, and efficiently cocreate and disseminate information with improved telepresence and mutual awareness. In that vein, the predominant user interface paradigm of WYSIWIS (*What You See Is What I See*), geared toward standalone work, has been relaxed [211] to account for differences in presentation of local and remote content. We refer to this paradigm as WYSIWISH (*What You See Is What I SHare*), indicating selective, fine-grained sharing of information within specific multicast groups and sessions.

In contrast to the majority of commercial and experimental CME existing to date, we look at collaboration as an inherently distributed process, where session coordination

and control are enacted collectively by participating hosts, rather than fixing such roles in centralized servers. We presents a formal model of group coordination and collaboration in networked multimedia systems and propose a turn-taking model, known from applied linguistics, to be used for analyzing the effectiveness of floor control protocols. With this model we are able to merge social- and machine-driven regulation of interaction in an integrated treatment of resource sharing. A comprehensive view on the components of group coordination motivates the work on floor control and ordered multicasting.

Floor control is the orchestration among users in multicast groups partaking in telecollaboration, deciding on who may access and act on a mutually exclusive resource at a given time. This notion of floor control for CME is a generalization of the established meaning of who may speak in an assembly. We provide a fresh view on floor control, in terms of a novel taxonomy and a comparative throughput analysis of the various paradigms existing to date, using a turntaking model known from psycholinguistics and insights from mutual exclusion, channel access, and multicasting. Our work on the network aspects of floor control differs from various other efforts in CSCW focusing more on social and human factors issues. In contrast to the many conceptual descriptions of floor control protocols in the literature, we provide specifications and verifications of two novel floor control mechanisms. We discuss design choices in implementing floor control protocols, integrating user concerns such as interfaces and interaction stages, and network concerns, such as locus and policies in the deployment of control, as well as Quality-of-Service.

Ordered end-to-end multicast guarantees that packet flows from different sources are received by all hosts in a receiver set in the same order, ensuring the collective integrity and consistency of distributed operations. This property is relevant for distributed applications depending on a consistent systemic view on collective activities, events and object updates at all participating sites, as it is the case for a shared whiteboard as an example for a distributed multiparty collaboration tool, or DIS systems. Few earlier research efforts on reliable multicast have investigated the ordering issue, possibly due to the end-to-end debate [39]. This debate revolves around the argument that ordering services should be

deployed at the application layer because lower layers lack knowledge of application-specific ordering semantics. We propose a novel taxonomy on reliable, ordered multicast protocols integrating known approaches from fault-tolerant systems and current networking research, and present a comparative performance analysis. It shows that our solution for tree-based ordering, which aggregates and orders messages en route from sources to receivers, is more efficient and less costly to deploy than previous solutions, in particular, when operating in conjunction with an underlying tree-based, reliable multicast protocol.

1.4 Dissertation Structure

This remainder of this dissertation is organized as follows:

Chapter 2 discusses principles of group coordination [51, 58], including a formal model for turntaking and a capability description to specify coordination sessions, providing a foundation for the chapters on floor control and ordered, reliable multicast.

Chapter 3 discusses concepts and related work [52, 53, 54, 55] on floor control, looking at end-user issues such as the interfaces and interaction stages, and Quality-of-Service tuning. A novel taxonomy and comparative performance analysis of known floor control protocol types [56, 57, 58] is presented, followed by specifications of two new floor control protocols. The *Floor Control Protocol* (FCP) [55] operates in networks without assuming a specific end-host connectivity, with the node controlling the floor roving among session members for reasons of efficiency and resilience. We motivate then the idea of hierarchical group coordination by discussing a multisite scheme operating on a tree topology [59, 60] in conjunction with tree-based reliable multicast. The *Hierarchical Group Coordination Protocol* (HGCP) [57, 58] implements this idea by performing floor control on top of a tree-based reliable multicast protocol for efficient aggregation and forwarding of control information among hosts, just as reliable multicasting uses the tree to efficiently relay error-recovery control messages without overburdening sources.

Chapter 4 investigates group coordination at the transport level, by addressing the problem of message ordering in reliable multicast communication. We first review related work and present a novel taxonomy for existing broadcast and multicast ordering solutions [61]. We elicit the advantages of staggered ordering of messages on the multicast delivery paths from sources to receivers, and present a specification of the *Tree-based Ordered Multicast* protocol (TOM), which incrementally orders messages from various sources on their paths to common receiver sets. In TOM, receiver groups may overlap and remain anonymous to senders. In contrast to previous solutions, no separate graph must be maintained for propagating ordering information, because TOM is designed to run on top of a tree-based reliable multicast protocol, using the underlying tree structure to communicate ordering information among sources, ordering nodes, and the receiver set.

Chapter 5 presents a summary of the contributions put forward in this dissertation and outlines a roadmap for future work.

Chapter 2
Group Coordination

Our objective for this chapter is to provide a comprehensive framework for group coordination from a modeling and architectural perspective, looking at the components of group coordination processes from various angles to better understand how it can be exploited for more efficient information routing and task execution. Group coordination in distributed systems and multimedia systems has many faces manifested in a variety of user interfaces and network protocols. To date, no standardized methodology for engineering group coordination protocols exists. We discuss various views on coordination, including insights from psychology and linguistics, to frame our technically oriented views. We discuss a formal model to characterize group cooperation environments and propose a turntaking model to characterize user interactions across computer networks. A group coordination architecture is motivated as a foundation for discussing two of its components, floor control and ordered multicast, in subsequent chapters.

2.1 Conceptual Foundations

With the advent of networked desktop computers, new opportunities arise to collaboratively utilize distributed information systems. Information processing is no longer limited to a singular, finite time-sharing domain. Client-server architectures have been a catalyst for new enabling technologies such as cluster computing, where groups of cooperating processors communicate seamlessly with each other. Computer-supported cooperative work, specifically multimedia telecollaboration, promises to deliver more powerful ways of cooperation

and coordination, compared to first-generation rudimentary examples of groupware such as email, data exchange, or online coordination of calendars.

Ellis and Rein [67] define groupware as "computer based systems that support two or more users engaged in common tasks, providing an interface to a shared environment". In their coordination theory framework, Malone and Crowston [145] define coordination as the "act of managing interdependencies between activities performed to achieve a goal", looking at components of actors (people) and agents (computerized procedures), identifying workgoals, mapping goals to activities, and managing interdependencies among actors and activities. They distinguish between generic interdependencies, for instance sequenced or simultaneous actions on shared resources, and domain-specific interdependencies, e.g., specific data elements that must be passed between team members to achieve successful groupwork. Schmidt and Simone [195] present an empirical characterization of computational coordination mechanisms useful as general blueprint for the design of coordination protocols, proposing for instance the construction of a mechanism such that "actors are able to control its execution and make local and temporary modifications of its behavior to cope with unforeseen contingencies".

Axelrod [17] investigated cooperation from a game-theoretic perspective, specifically, how the tradeoff between individual greed and good affects coordinative strategies in groups, assuming rational behavior. This problem is also known as the *social dilemma*. Focusing on "tit-for-tat" games, interactions are interpreted as pairwise alternations of moves with specific payoff values. The pay-off structure of the interaction determines the game motive. For two participants A and B, the payoff structure for choosing two actions i and j is $P = A_{ij} + B_{ij}$. If $P = 0$, then the interaction is called a *zero-sum* game, and interactions with $P \neq 0$ are called cooperative or mixed-motive games.

A related approach uses economic models to tackle resource allocation in computer systems from a market-oriented perspective [75]. A cost function is assigned to cooperative activities, individual negotiations, deals, and strategies. An activity between two subjects is *pareto-optimal* if it is not possible to improve the utility for one subject without lowering

the utility of the other subject. A strategy to determine the progress of activities is said to be in equilibrium if no party has an incentive to diverge from that strategy in order to fulfill individual and group tasks. Multiple equilibria are possible and two strategies S and T are said to be in *Nash-equilibrium* [154] if one party cannot do better other than using T, when the other party uses S, i.e., the product of individual utility values is maximized. However, it is not simple to assess global utility values, and choosing one of several possible equilibrium points may guarantee relative fairness but restricts the space of possible agreement states, under the assumption that all subjects employ the same utility measure and do not cheat. ContractNet [208] was an early market-based protocol approach towards distributed task-completion, employing a bidding scheme among managing and contracting nodes. Shenker [204] argues that applying a fair-share service discipline at network switches models uncooperative flow control satisfying individual users' selfishness more realistically than traditional disciplines, which presuppose cooperation such as First-Come-First-Served among users.

Cooperation has been a major research focus in distributed artificial intelligence. Lux *et al.* [142] define a language of cooperation between humans and agents to assess how humans and computational agents estimate payoff in resource usage and relinquishment to optimize their strategies. Rosenschein and Zlotkin [190] developed a framework on negotiating protocols among agents, where utility metrics define strategies and outcomes in cooperative resource allocation. Decker [46], understanding coordination as the "act of managing interdependencies between activities", dissects coordination into specification of shared goals, solution planning, and task scheduling, starting with signals for attention, acceptance or refusal of a connection, information transfer, acknowledgment, and teardown. Blackboard architectures [105] have been extensively studied in artificial intelligence as a mechanism for sharing information among expert system modules for purposes such as multiple-task planning.

Holt [109] discusses a coordination language based on Petri nets, as an element in a larger theoretical framework referred to as coordination mechanics, to establish relationships

between tasks and products. Winograd [234] presents an analysis of group action based on speech act theory and discuss the COORDINATOR tool to support users in keeping track of requests and commitments to each other.

Rabin [177] defines *choice coordination* as the problem of designing a wait-free protocol for n concurrent processes causing all correct processes to agree on choosing one out of k possible alternatives. Communication is performed via registers associated with choices, and the question is: how can choice coordination be best solved deterministically. If a small probability of failure is acceptable, a protocol on a small alphabet can solve the problem; however, a "perfect" protocol is very expensive. Joung and Smolka [119] proposed a taxonomy of languages describing multiparty interaction, discerning between interactions with fixed vs. variable participants, conjunctive vs. disjunctive parallelism in the execution of interactions, synchronous vs. asynchronous task execution, and biparty vs. multiparty scenarios. For synchronous systems, multiparty interaction with disjunctive parallelism is accordingly NP-complete, and biparty interaction is solvable in polynomial time, when using conjunctive and disjunctive parallelism in isolation.

Coordination can be manifested explicitly, e.g., through signals like acknowledgments, or implicit through interpretation of the behavior and responses of partners. The process of coordinating activities can hence be transparent or tangible to applications. Coordination revolves around resource allocation in terms of scheduling and availability. Allocating a resource at time t may create contingencies for resources and users at time $t + 1$. Coordinative stages are therefore often non-separable. Messaging may be based on ad hoc input or use a predefined vocabulary. Partners may be located at the same or different places, and coordinate at the same or different times, i.e., coordination processes may operate synchronously or asynchronously in the relative timing of partners. Participants at any instant are either active initiators, or passive receivers. Coordination flow may be unicast and either half-duplex or full-duplex, or multi-way, using broadcast and filtering, or multicast.

Coordination is intertwined with conflict [111], and its prediction, avoidance, or resolution. Conflict can arise before the allocation of shared resources, implying contention

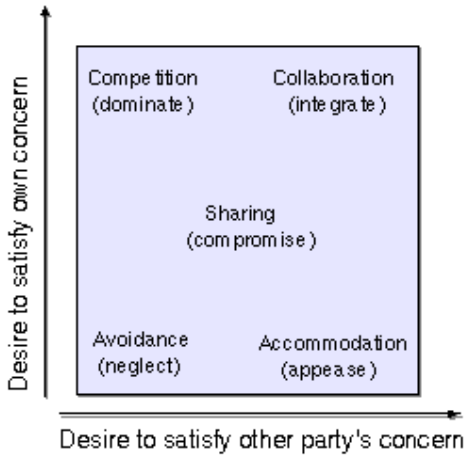

Figure 2.1: Conflict resolution in terms of own vs. group concerns.

or race conditions, or during and after resource allocation, creating collisions, congestion, inconsistencies or information loss. Conflicts are resolved by blocking or queuing contending processes or by aborting them. A specific conflict prevention or resolution scheme may be unfair in that it benefits specific users and neglects others. Figure 2.1 [64] depicts the strategies of conflict resolution in the light of personal vs. group interests. We can partition the set of N users into two subsets, cooperating and conflicting users $\{u_i\} = \{u_j\} \cup \{u_k\}$, for i, j, k $= 1, \ldots, N$ and $j \neq k$. The objective of group coordination is to minimize the conflict set, e.g., by establishing effective turn-taking protocols.

2.2 Formal Framework

In this section, we present a formal view on entities and actions refining earlier efforts [180, 181] on the definition of coordination and control processes in collaborative multimedia systems. Candan *et al.* [32] focus on algorithms for collaborative composition and transmission of media objects under given quality constraints, and their presentation in collaborative group-sessions. We picture a computer network as a graph with nodes (stations, hosts) V sending messages across links (channels) $E \subset V \times V$. A connection is a unidirectional or bidirectional transmission link from a sender node to a set of receiver nodes.

Definition 1 *A collaboration environment* Γ *in a computer network is a tuple*

$$\Gamma =< \mathcal{S}, \mathcal{U}, \mathcal{R}, \mathcal{F} > \tag{2.1}$$

where $\mathcal{S} = (V, E)$ *is a set of sessions* Σ, $\mathcal{U}$ *is a set of users (hosts, processes, agents, participants),* $\mathcal{R}$ *is a set of shared resources (media), and* $\mathcal{F}$ *is a set of floors controlling the resources.*

2.2.1 Entities

Sessions

A session provides the infrastructure for cooperation and collaboration.

Definition 2 *A session* $\Sigma \in \mathcal{S}$ *is a tuple*

$$\Sigma =< Sid, T_i, T_e, A_S, L > \tag{2.2}$$

where Sid *is a unique identifier within* Γ, T_i *is the initiation or announcement time,* T_e *is the ending time, and* A_S *is a list of attributes characterizing the session at level* L. *A conference is a set of sessions* $\Sigma_i \in \mathcal{S}$, *where* $i \geq 1$.

Sid is a unique session identifier per collaborative environment, whose sequence number space is wrapped around in correlation with the turnover rate and lifetime of sessions in Γ. The time may reflect real-time, logical time, or define a lifetime interval $\Delta = T_e - T_i$. L denotes the session level (default 0).

$A_S = (M, O, C)$ describes purpose and orchestration of a session in terms of membership M, organization O, and control C, as shown in Figure 2.2. Szyperski [216] characterizes session types in a similar, but less refined way, according to the model of interaction (controlled, dynamic, static) and data flow ($1 - n$, $n - 1$, $m - n$). For instance, a lecture is a controlled, long-term interaction between one sender and n receivers. Telemetry is a typical $n - 1$ session, and a whiteboard session is typically $m - n$. Our session characterization

applies to specific collaborative applications, as well as generic session types in the spectrum of real-time collaborative work, such as lectures, business meetings, labs, panels, brainstorm meetings, exams, interviews, or chats.

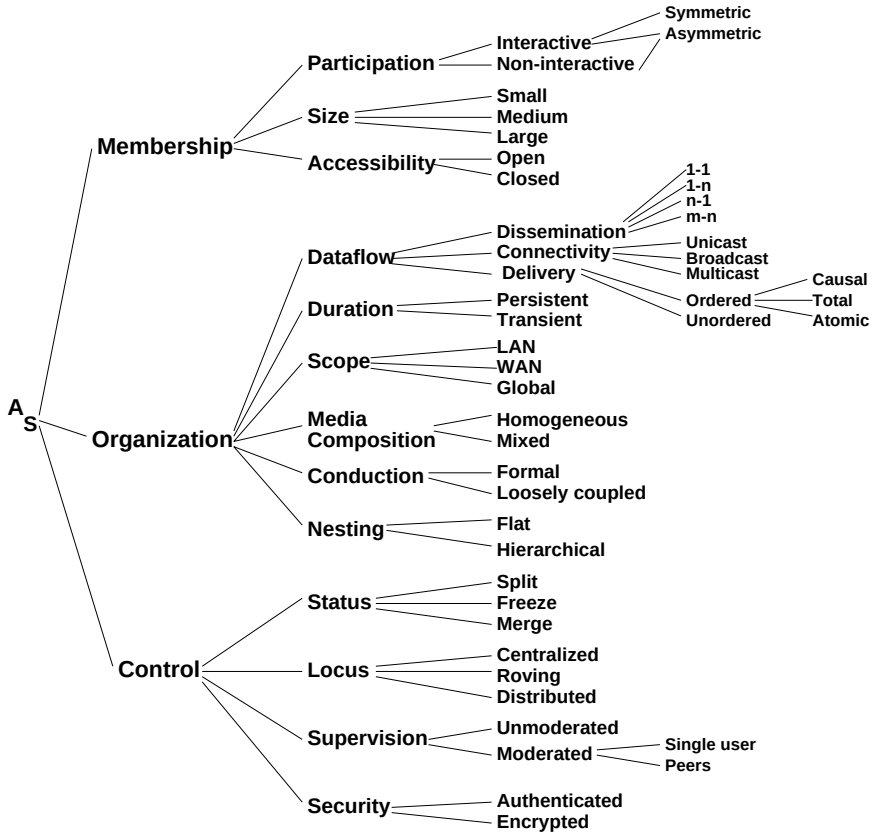

Figure 2.2: Session attributes.

Membership reflects the composure of the user group in the session. *Participation* specifies whether information is exchanged unilaterally, or bilaterally relative to a host, impacting user access rights and data-flow. Interactive sessions may be symmetric, i.e., all users have the same view on shared resources (WYSIWIS), or asymmetric, where users pertain individual views on the same shared data space (relaxed WYSIWIS) [211]. *Size* specifies a small (< 5), medium (< 100), or large (≥ 100) number of users, impacting scalability of the coordination mechanism. *Accessibility* declares whether a session is open, allowing any user to join, whereas closed sessions allow participation by invitation only. Authorization specifies

whether coordination primitives may use read-only, read-write, or write-only privileges for the entire session. Users may have individual, role-based authorizations, as well.

Organization entails specifics on how the session is to be orchestrated. *Dataflow* describes how data are multiplexed among users, with a $1-1$, $1-n$, or $1-m$ transmission model and with unicast, broadcast, or multicast in a session of n users, where $m \leq n$. Delivery can be ordered or unordered. The various ordering modes are discussed in Chapter 4. *Duration* discerns between sessions with longer lifetime (persistent) vs. short-term sessions, where the precise timing modalities are case-specific and left open. The *scope* specifies the hop limit for packets sent by hosts in a particular session, similar to the Time-To-Live semantics in IP, which allows constraining sessions to a geographic range and retain privacy or limited dissemination to a specific group. *Media composition* defines whether the session uses a single medium such as audio-only, or mixed media, e.g., a video-audio combination. *Conduction* refers to the session agenda and moderation style, which can be either tightly coupled, i.e., all users know about each other and follow some agenda in the style of "Robert's Rules of Order" [185], or the exchange is loosely-coupled and not prescribed. Sessions can be flat ($L = 1$) or maintain two or more levels with *nested* groups ($L > 1$).

Control depicts the status, locus of control, and security measures activated for a session. Sessions with overlapping or diverging interests can merge or split. Such reconfiguration of sessions with regard to membership and session events linked to specific phases must be possible without session termination or restart of applications. The session *status* marks whether the session is a partition from a larger session, frozen but still deemed as active, merged or revived. Tracking of states in coordination protocols and the outcome of coordination processes can be logged and persistent, or ephemeral.

Locus of control specifies, whether membership and floor control are being handled in one central location, partially distributed among several servers, or fully distributed across all hosts. Partial or full replication is possible for the latter two paradigms. A central controller can also rove among all sites and achieve better fault tolerance. Distributed control is

multilateral, with varying degrees of "consentience" and "equipollence", i.e., how much everybody participates and how authorities and responsibilities are allocated. Multilateral control is either successive, partitioned, democratic or anarchic. Successive controllership allows one distinct controller at a time, and alternates among users, and partitioned control lets several controllers each perform a subset of control operations. Democratic control lets all users contribute to the control process, e.g., via voting. Anarchic control gives all subjects complete freedom of acting and control of sharing is peer-to-peer based.

The locus of control is related to the *supervision* attribute, indicating whether the communication process in coordination is moderated, peer-reviewed, or free. A moderator decides which users may send information, what is forwarded to the receivers, or which receivers may receive a particular content or access a specific resource, implementing a notion of floor control. McKinlay *et al.* [151] note for face-to-face meetings that the importance of chaired guidance increases with the session size, and the difficulty in performing a joint task, since each member's ability to participate and influence others is reduced. Finally, coordination touches upon *security* issues, specifying whether users are anonymous or authenticated in their exchanges, either at session initiation, or at every turn, and whether information is encrypted.

Hierarchical Sessions

Rajan *et al.* [180] identify a *confluence* as a special session type, where all participants transmit and receive the same set of media streams mixed together in broadcast, which saves bandwidth. The notion of confluences and session nesting leads to the concept of multilevel or *hierarchical* sessions. Session hierarchies permit aggregation of users at various levels of abstraction, reflecting interests, the stage of task completion, authorizations, temporary subgroups (coteries), or geographic proximity, and mirror the inherently hierarchical group dynamics of face-to-face meetings. The hierarchy is denoted with the session level parameter L, which indicates numerically the position of *Sid* in a session hierarchy. For instance, in a 3-level hierarchy, a collaboration or master session has level 0, a session level has level 1,

and a subsession is at level 2, which may be sufficient to characterize most collaboration scenarios.

Rangan and Vin [181, 225] give formal definitions for collaborative systems including conference, session and stream abstractions for the purpose of automated reasoning about the properties of multimedia collaborations. Adopting their definitions to the session context, we distinguish between *simple sessions* containing individual users, and *super sessions*, recursively consisting of other sessions $s_{i,L} \in S_i$ and individuals, with L indicating the level of membership. We denote the outmost "root" session as level-0 session. Many conference scenarios contain only two sublevels, *subsessions* with $L = 1$ and *coteries* with $L = 2$. Coteries permit private subgrouping for brief exchanges ("sidechats") [227] without requiring its members to leave the larger group context or open a separate multicast group. Neilsen and Mizuno describe a membership algorithm for joining and leaving coteries [157], and Texier and Plouzeau [218] propose object binding algorithms for multiple sessions, however, to date a sound mechanism for session management in multimedia collaboration is still missing.

We also adapt the notion of *concurrent* sessions, as opposed to sequential sessions, which allows users to participate in multiple sessions simultaneously. Furthermore, *hierarchically-related sessions* allow for inheritance of attributes from parent to child sessions, and aggregation of sibling sessions under a parent session. Figure 2.3 depicts a simple session graph with three concurrent sessions S_1, S_2, S_3 within a conference C, two subsessions $s_1, s_2 \in S_2$ and one coterie $c \in s_2$, containing users U_2 and U_5 using video. Clustering of users, groups and sessions may be based on task focus, geographic proximity, media compatibility, and other criteria. Except for S_3, all other sessions are chaired, which is symbolized with a rhombus. As depicted in $s_1 \cup s_2$, media should be sharable across session boundaries and users can actively be part of two or more sessions at the same time.

This session structure can also be represented in list notation, separated by the media in use. Floor holders, a system role defined in the next section, are denoted in boldface:

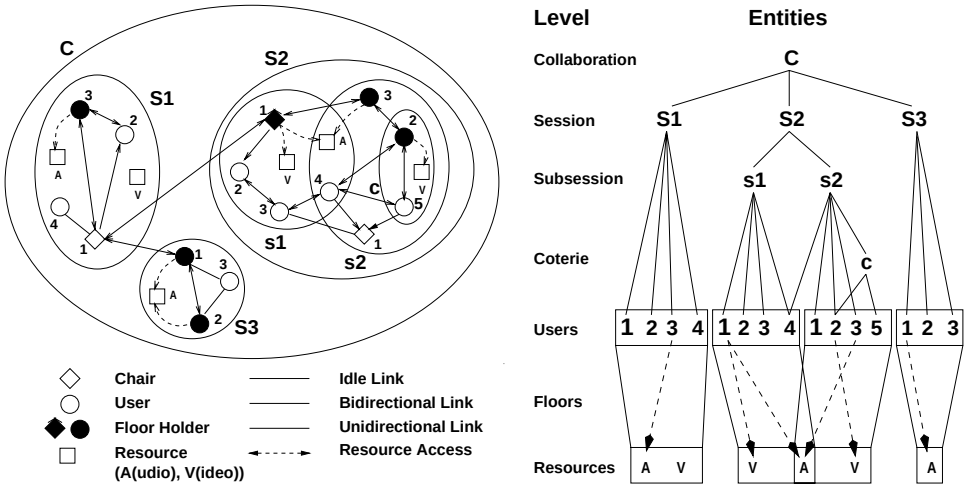

Figure 2.3: Hierarchical aggregation of concurrent sessions and corresponding session graph.

$$\begin{aligned}&((12\mathbf{3}4)^{S1}((\mathbf{1}234)^{s1}(1\mathbf{3}(25)^{c})^{s2})^{S2}(\mathbf{12}3)^{S3})^{C}_{A}\\&((\mathbf{1}234)^{S1}((\mathbf{1}234)^{s1}(13(\mathbf{2}5)^{c})^{s2})^{S2}(123)^{S3})^{C}_{V}\end{aligned} \tag{2.3}$$

Let the relation $(S_i \tau_{ij} S_j)$ express a temporal relation (e.g., `starts, contains, overlaps, split, merge, end, info`), given sessions S_i, S_j, with i, j natural, $i \neq j \in \mathcal{S}$. Then

$$\mathcal{S} \supseteq S \supset s_L \supseteq c \tag{2.4}$$

Vin *et al.* [226] describe such a hierarchical architecture for media mixing, as required in a telepresentation system forming a teleorchestra, and derives upper bounds for the media transmission capacity and the height of a hierarchy, given a number of participants and mixers, with one speaker being active at a time.

Users

Users belonging to the set $\mathcal{U}$ in our formal specification of a collaborative environment are also referred to as participants, subjects, or session members. Alternatively, a user is equated in protocol descriptions with the host or the processes in use. We define users and their roles as follows:

Definition 3 *A user $U \in \mathcal{U}$ is a tuple*

$$U =< Uid, Sid, Loc, T_j, T_l, A_U > \tag{2.5}$$

where Uid is a unique identifier within the session Sid, Loc is the local or remote location, given as IP-address or unique host identifier, T_j is the joining time, T_l is the leaving time, and A_U is a list of user attributes.

Processes can be system agents [65, 134] executing on behalf of a user. The user attributes A_U are depicted in Figure 2.4.

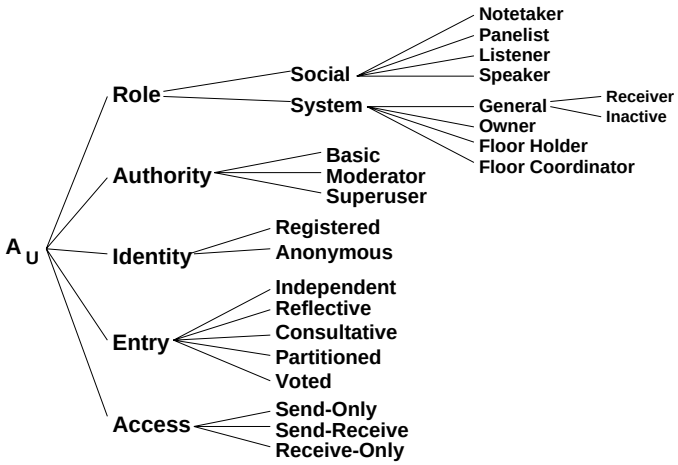

Figure 2.4: User attributes.

Accordingly, users are characterized by their roles, authority, identity, entry capabilities and access rights, which impact the applicable floor control strategy. Users can be co-located in the same space, or geographically distributed. We distinguish between social and system *roles*. *Social roles* describe the function of a user within a session, e.g., being a panelist or lecturer. *System roles* refer to the control function within a floor control protocol: participants without a specific role can be either a receiver or inactive. The owner of a resource R is the node that injects R into a session and initiates floor control for R, which may vanish from a session if the owner leaves. The floor coordinator (FC) is an arbiter over a resource R, or a session moderator granting or denying a floor on R

during session time to the floor holder (FH), who attains the exclusive right to work on R for a floor holding period. FC and FH may be located at different nodes, or be assumed by the same node. These roles may be statically assigned at session start, or rove among users during session conduction. Users without control roles are general session members, and can be active or inactive, depending on whether they invoke state transitions in the coordination mechanism.

A moderator is a special FC case, where the coordinator role is assigned to a user to supervise content exchange, resource usage, and membership for a specific section of the full lifetime of a session. A moderator-driven session, mediated through a specific host, results in a centralized coordination scheme, even though the host topology may be decentralized, with known shortcomings in efficiency and resiliency. Moderators may be selected, because they start a session or are chosen by session members in advance, or they may be elected [85, 207] at session runtime. Tijdeman [219] discusses a solution for the chairman assignment problem such that at any time the accumulated number of chairmen from each state (or session) is proportional to its relative weight. Role-based floor control in dynamic sessions contrasts static *role-based access control* (RBAC) models [193]. Roles can be inherited from a supersession to a subsession.

Authority defines whether the user is a simple participant, privileged as system root user, or moderator, linking this field with the role entries. A moderator can be permanent FC. As a social role, the moderator equates to a session supervisor being able to inspect all session turns between users. *Identity* specifies whether the user wants to remain anonymous or whether the Uid can be posted to the session. An *entry* is either independent, i.e., unaware of the actions and entries of others, reflective, i.e., polling session members, consultative based on "contextual clue messages", partitioned and representing a subtask, based on voting among the group, or debriefed and recorded [67]. In addition, user entries may be temporary or permanent, and logged for the purpose of reviewing histories of collaborative sessions, or for undoing certain steps [176]. *Access* defines the basic privileges for working on a resource, in receive-only, send-and-receive, and send-only mode, analogous to read

and write authorizations in file systems. We introduce the notion of a group to describe associations of users within sessions.

Definition 4 *A user group (multicast group) G is a set of users U with common session and user attributes, expressing a common media or task focus, such that $\mathcal{U} \supseteq G \supseteq U$.*

Resources

Multimedia collaborative systems use a polymorphic or multimodal mix of resources being shared across networks. A resource can be an application, host object, or network entity shared in collaboration at various levels of granularity. Four primary classes of multimedia traffic with different Quality-of-Service characteristics exist [5]: *control packets* for coordination information are mostly of low volume, but need reliable transmission; *real-time media* transport time-critical information and tolerate some loss; *elastic media* are apt for discrete information with relaxed timing constraints, but tolerate no loss; and *bulky media*, which require high throughput and reliable transmission, but can tolerate some delay. We define resources as application components in our coordination framework:

Definition 5 *A resource $R \in \mathcal{R}$ is a tuple*

$$R =< Rid, Sid, Pid, Uid, T_c, T_d, A_R > \tag{2.6}$$

where Rid is a unique resource identifier owned by user Uid within session Sid. Pid is the parent identifier or the resource that Rid belongs to, T_c is the time of creation or injection of the resource into the collaborative workspace, T_d is the deletion time, and A_R is a list of resource attributes.

Rid designates both discrete media and streaming media and may contain the port where the resource is transmitted. The resource attributes A_R are depicted in Figure 2.5. The Pid value allows for recursive subsumption of resource components within resources, and hence sharing or resource components at an arbitrary granularity. For instance, users can share an entire window, or a graphical object within that window.

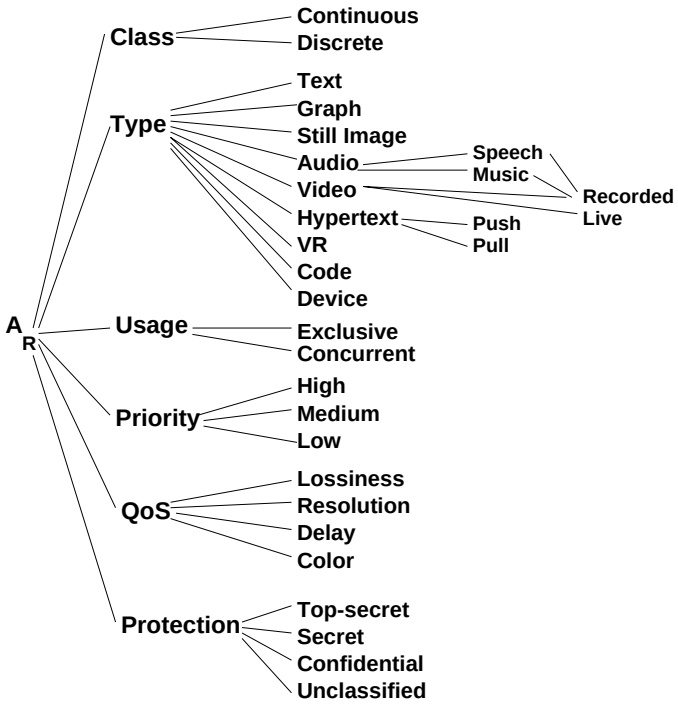

Figure 2.5: Resource attributes.

Class describes whether the resource is continuous or discrete. *Type* characterizes the media object class, indicating whether a resource is text-based, graphical, or some real-time medium and identifies the purpose it serves. Resources can be virtual, or they can represent actual remote devices, for example, a surgical instrument in telemedicine. Resources can be mixed and need not necessarily be proprietary to the session from which they are accessed, but could also be hosted on a machine "outside" of the session. Coordination on *text*, as the default medium for most collaborative system, revolves around alternate typing, for instance in chat tools, or concurrent editing from chapters or sections to single sentences. Text can be plain ASCII, or one or various rich text formats with formatting commands. *Graphics* tools, such as drawing and design tools, necessitate coordination in time and space, either by marking areas on a shared canvas or objects for shared editing, or by introducing graphical widgets such as telepointers. Functions that compute or render the shared workspace in a specific way are another coordination component. *Still images* require also spatiotemporal coordination and allow for multiple image formats, such as TIFF, GIF, or JPEG.

Audio tools, for speech, or music data (such as MIDI) require temporal coordination in recording and replay, and spatiotemporal coordination in editing. For instance, a shared audio channel or music stream requires sequenced access, whereas joint editing of a music score is a spatial aspect. Silence detection is useful for more efficient processing of audio streams, but also help to trigger speaker floor switching. *Video* concerns motion image display and editing, either from a live source, stored locally, or replayed on demand, and is often used in combination with audio, requiring temporal coordination. Various formats, such as H.263 or MPEG, would be supported. *Hypertext* information is multimodal and integrates all of the above resource types using, for instance, HTML or XML, and is either geared for server-push or client-pull. *VR* (Virtual Reality) [38, 92] is similarly multimodal, but adds input and output devices giving the user three-dimensional orientation or tactile sensations. Coordination must be interfaced with collision control [110] in virtual spaces. *Code* comprises application-specific structured documents such as Postscript, MIME email [28], or LaTeX. A *device* is a hardware unit serving as access point, such as a camera. A *multimedia conference* is a conference using multimodal resource types.

Usage determines if the resource can be used concurrently by multiple users or requires sequential processing with exclusive floors. For instance, a shared whiteboard allows for multiple concurrent telepointers with a small number of users, whereas a remotely controlled camera can only perform a positioning command for one user at a time. *Priority* sets an importance value on the transmission and processing of the information, preempting other media dissemination of lower ratings. *QoS* defines the required Quality-of-Service [213] for the resource, including the tolerable loss, the required resolution, the possible maximum delay, and the color depth. Other criteria may be added depending on the nature of the resource, such as the channel number, a frame-rate, encoding scheme, sampling rate etc. The *Protection* attributes denotes whether a resource is public, private, or proctored, which may be expressed with a numerical value, or work in analogy with the *Bell-LaPadula* model [21], discerning between top-secret, secret, confidential, or unclassified information [69]. The degree of security determines the required encryption level and method to prevent forgery

of control states and coordination messages. In contrast to traditional models of protection giving access to a resource based on user identity, coordination-based access must take into account the task to be performed. Predominant measures to shield off internetworks with firewalls make real-time collaboration very difficult and are a major impediment in the realization of Internet collaboration. While new concepts for secure collaboration architectures are emerging [88], efficient key management and encryption in conjunction with floor control have yet to be developed.

Type	T	S	RT	L	J	BW	FD
Text							
Editor		√				l	t, s, f
Chat	√	√				l	t
Email						l	t, f
Scheduling		√				l	t, s
Coding		√				l	t, s, f
BBS/Usenet						m	t, s, f
Spreadsheet		√				l	t, s, f
Audio							
Speech	√	√	√	√	√	m	t
Sound	√	√	√		√	m	t, f
Images/Video							
Still				√		m/h	s, f
Motion	√	√	√	√	√	h	t, f
Graphics							
Still $\{2D, 3D\}$		√				m	t, s, f
Motion $\{2D, 3D\}$	√	√	√			m/h	t, s, f
WWW		√			√	l-h	t, s, f
Virtual Reality	√	√	√		√	h	t, s, f

Table 2.1: Resource types and handling characteristics.

Various collaborative applications use different mixes of resource types: chat tools are text-based; multimedia email can contain text, images and sounds in various file formats; scheduling tools are typically text or graphics based; collaborative computer-aided software engineering (CASE) tools are typically text-based; shared spreadsheets are text-based, which includes numerical entries; audio tools can use continuous or stored soundstreams, without or with known time limits. Similarly, music tools, using for example the Musical Instrument Digital Interface (MIDI) format, use specific sound encodings; still images use specific encodings such as JPEG; video tools use specific motion video encodings such as

MPEG or MPEG-2; hypertext systems such as World Wide Web browsers use any mix of text, images, graphics, audio and video; and virtual reality systems use graphical and photo representations in combination with sound, encoded in descriptive languages such as VRML. The media types used by these generic applications determine their control and coordination characteristics. Media types can be: lossy (L) vs. lossless; transient (T) vs. persistent; are suited for synchronous (S) or asynchronous groupwork; may require real-time transmission protocols (RT); can be vulnerable or insensitive to latency (L) or jitter (J); and may require high or low bandwidth (BW). Media with focus on spatial sharing may require spatial (s) floors (F) for access regulation, in contrast to media, where contention arises in the temporal domain (t), or with regard to the use of specific functions (f). Table 2.1 summarizes these media properties (a $\surd$ indicates that the property applies for the respective media type).

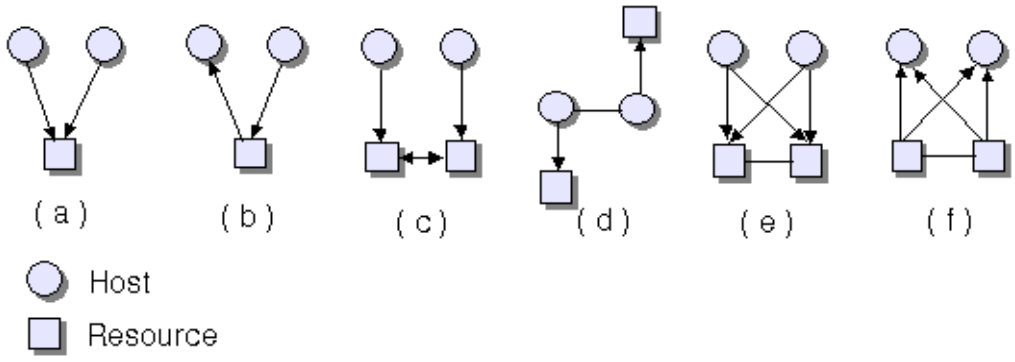

Figure 2.6: Resource access scenarios: (a) Centralization, (b) Producer-Consumer, (c) Replication, (d) Distribution, (e) Multi-resource access, (f) Multi-resource consumption.

Resources $r \in R$ can be located at one particular node, or distributed in their components across the node set, or they can be replicated at several or all nodes. Figure 2.6 depicts the various access paradigms for shared resources. In case (a), one or more resources are centralized and accessed by multiple parties; case (b) lets one host produce a resource and other hosts consume it; (c) shows the case where each party maintains some replica of the same resource locally, exchanging updates on a regular basis; in case (d) all hosts maintain partial information on the shared resource, using a distributed protocol to aggregate

the information; and cases (e) and (f) show access or consumption of multiple resources by multiple parties. These constellations are the baseline for configuring a coordination mechanism to adapt to various configurations of the shared workspace. A location mechanism for resources within sessions, and mapping scheme from resource objects to multicast groups is needed, as partially implemented with the CCCP protocol [104].

Floors

A floor is a temporary access and manipulation privilege for multimedia resources in interactive groupwork. It is a computational metaphor for the "right to speak" [185], generalized to the domain of CSCW. A floor control protocol mediates access to shared objects by granting floors according to a group-specific service policy.

Definition 6 *A floor $F \in \mathcal{F}$ is a tuple*

$$F =< Fid, Rid, Uid, T_i, T_d, A_F > \tag{2.7}$$

where Fid is a unique floor identifier within the shared workspace for a resource Rid, assigned to user Uid at inception time T_i, and deactivated at time T_d, with A_F denoting a list of floor attributes.

Note that one *Rid* may have multiple *Fid* assigned for control of various granules, but each floor is controlling exactly one resource. Floors are indirectly associated with sessions via *Rid*, and floor properties may be inherited from a master resource to its subcomponents. We assume that one floor F is assigned per resource component. The pairing (*Fid*, *Rid*) specifies the granularity of control and the commands available with possession of the floor. A floor can control an entire conference, an application, a single window, or a shared object [133]. For instance, for audio the associated commands may be `talk, mute, pause`. Video floor commands are for instance `caption, forward, cut, replay`. Floors can be static relative to a session lifetime, or dynamic, i.e., assigned ad hoc by a computer or social protocol. The combination of *Uid* and the attributes specifies whether the user is *FC*,

FH, chair, or general participant. T_i and T_d may be set using real-time clocks, or a logical session time. Figure 2.7 depicts the floor attributes.

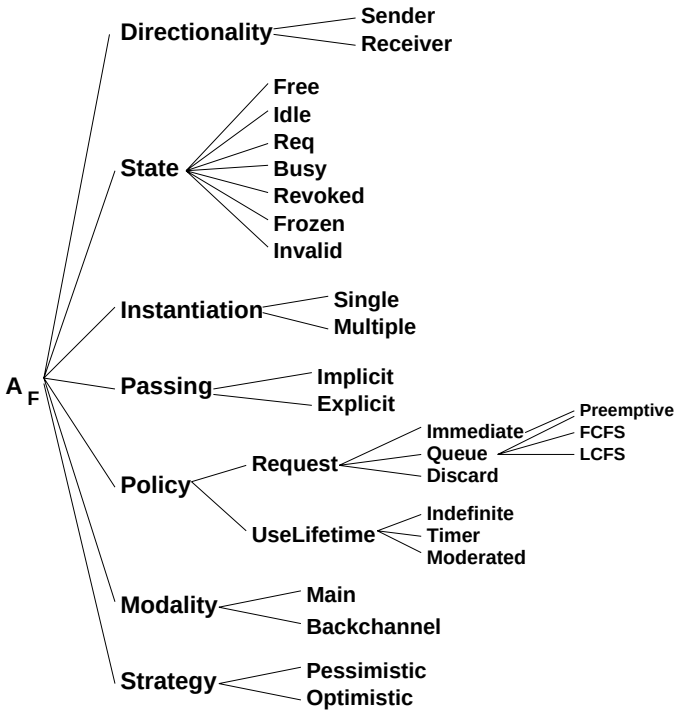

Figure 2.7: Floor attributes.

With regard to *directionality*, we discern between *sender floors* and *receiver floors*. A receiver floor refers to the passive control concept that enables a user to filter or deny specific received streams ("What I See Is What I Want"). Floor control typically refers to sender-oriented control, which may reduce traffic significantly ("What You See Is What I Share"). *State* defines the generic operational states of a floor control mechanism. *Free* denotes an available, unused floor, *Idle* denotes an assigned, but inactive floor, *Req* marks a floor as being requested, *Busy* is the tag for a granted and assigned floor. A generic floor control protocol defining the transitions between these states is depicted in Figure 2.8. Additional states can be introduced, e.g., *Revoked* marks a floor, whose lifetime is shortened by a moderator or a preemption mechanism, *Frozen* marks a floor in a pending session, and *Invalid* identifies a nonexistent floor.

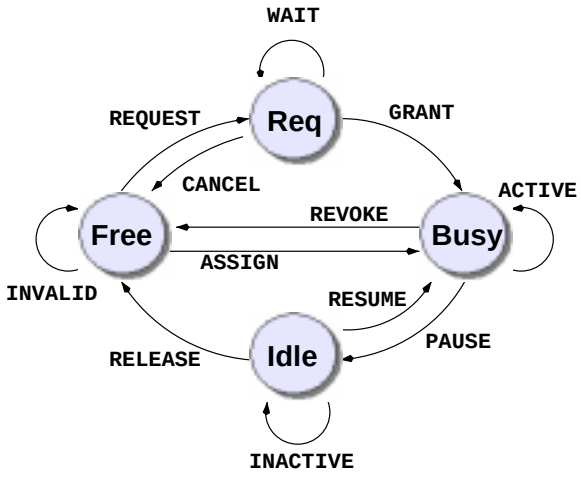

Figure 2.8: Generic floor control protocol.

Floor control can be relaxed for concurrent activities where the chance of direct conflict is smaller, e.g., in joint editing of text paragraphs, but it must be strict in opposing activities such as speaking over the same audio channel. *Instantiation* defines, how many instances of the same floor may exist concurrently in the system. A remote instrument with exactly one function to be shared permits a *single* floor, whereas telepointers on a whiteboard canvas may coexist in *multiple* renditions. Disjoint parties may receive multiple instances of a floor, e.g., user groups $(U_1, U_2)_A$ and $(U_3, U_4, U_5)_A$ may independently converse with an audio floor $F = A$.

Passing describes whether floor management is tangible or transparent to end-users. *Explicit* control gives handles to users to start and initiate turns based on the exchange of markers that signify possession of the floor, in contrast with *implicit* control, where no beacons are exchanged to transfer floors. Control may follow a programmed session agenda, or allow for open interaction. Explicit control is manifested for instance by pressing the `Request` button in a shared application. Implicit control is realized by users observing inactivity on the resource and taking action when appropriate. *Policy* will be discussed in more detail in Chapter 3 and defines the request and usage rules. The *request* policy determines whether floor requests are immediately satisfied, queued and served according a queuing policy, or discarded, when there is no opening for the floor. A chairperson may preempt any floor activity. *UseLifetime* denotes whether a floor can be used indefinitely

until being requested, or whether a timer or moderator controls the duration of usage. *Modality* distinguishes between *main* floors assigned for primary communication from a sender to a receiver and *backchannel* floors used to give brief feedback.

We can distinguish between four paradigms to deal with race conditions in cooperative work: *blocking* of conflicts with exclusive locks, *disallowing* of conflicts with permission tokens, *mitigating* conflicts by detecting dependencies and reordering of activities into non-conflicting series, and *resolving* of inconsistencies created through conflict. The first two paradigms are restrictive and prevent conflicts, the latter two are permissive and allow for progress into conflict with preconditions and postconditions. Therefore, the *strategy* entails *pessimistic* control following the premise of conflict avoidance, versus *optimistic* control as the strategy to allow conflicts and provide means such as dependency detection [211] to resolve them.

Several other notions are important to the concept of "controlling the floor". For facilitation of some sessions, a *chair* provides the necessary group cohesion, moderating activities and hence ensuring rapport among members. *Assistive* floor control aids a session chair in orchestrating a session, while *autonomous* floor control steers sessions without chair guidance. Floor management can be *manual*, supporting a human conference chair to determine which user in a session may speak next, or in a larger sense, become active on a shared resource. *Automatic* control lets users gain access to shared resources, steered by a floor control process (floor "daemon", "agent", "engine"). Such a module may be intergrated in the applications, or support applications as middleware. We furthermore distinguish between *temporal* (t), *spatial* (s), or *functional* (f) floors designating the control domain needed to resolve in groupwork with a specific resource type. Temporal sharing indicates timely or causal conflicts, e.g., in the alternation of speakers in conversations. Spatial sharing conflicts arise by using the same presentation area or storage space of a shared workspace, e.g., pixel areas in a drawing canvas, a fixed-size buffer or paragraphs in a text document. Functional sharing centers around usage of the same application functions that change the status or content of a shared resource, e.g., `zoom in` for remote camera control.

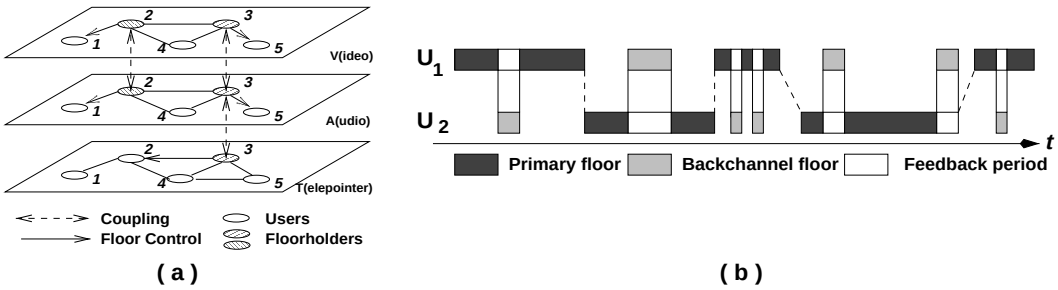

Figure 2.9: (a) Concurrent, coupled floors, and (b) backchannels.

We furthermore distinguish between *mutually exclusive* and *mutually selective* floor control schemes, where either one floor is assigned to n users on one resource, or k floors are assigned to n users, $k \leq n$. While exclusion avoids write-conflicts, selection relaxes control, but does not guarantee avoidance of conflicts, e.g., with the concurrent use of telepointers. Figure 2.9 (a) depicts a scenario where users communicate with video, audio, and telepointers. User pairs (U_2, U_1) and (U_3, U_5) do not overlap in their usage of audio and video. Dual assignment of *concurrent* floors within the same group is hence possible. This concept is similar to shared locks [3] in database systems, where two-phase locking is extended by allowing two locks to be held concurrently by different transactions to execute on the same object. Floors may be *coupled*, i.e., jointly assigned for the combined use of media R_i, denoted as $\oplus[U, (R_i)]$. In Figure 2.9 (a), we have $\oplus[U_2, (V, A)]$ and $\oplus[U_3, (V, A, G)]$, representing the joint use of telepointers with audio, as for instance in a lecture. Coupling of floors necessitates synchronization mechanisms for mixed media [170]. The concept of backchannel floors usable for short-term feedback is depicted in Figure 2.9 (b).

2.2.2 Capability Description

The presented model serves both theoretical and practical purposes. It provides a more elaborate framework for formal specification and validation of collaborative systems, e.g., with the prototype verification system [180]. It also allows for session capability descriptions [166] to set up and query the membership and coordination status of an active conference, where a capability is understood as a resources or system feature influencing the selection of useful configurations for components.

```
<C> = <S> <U> <R> <F>
<S> = (2374, 1700, 2000, <M>, <O>, <C>, 0)
<M> = (noninteractive, small, closed
<O> = ((1-n, broadcast, total), transient, LAN, mixed, loosely_coupled, flat)
<C> = (normal, centralized, single moderator, authenticated)
<U> = (1, 2374, local, 1700, 2000, <Ro>, <Au>, <Id>, <Et>, Ac>)
<Ro> = (speaker, FH)
<Au> = (moderator)
<Id> = (registered)
<Et> = (independent / consultative)
<Ac> = (send-receive)
<R> = (23, 2374, nil, 1, 1700, 2000, <Cl>, <Ty>, <Us>, <Pr>, <QoS>, <Pr>)
<Cl> = (cont)
<Ty> = (live video)
<Us> = (concurrent)
<Pr> = (high)
<QoS> = (10%, 352 x 288, 15fps, low, rgb)
<Pr> = (unclassified)
<F> = (12, 23, 1, 1700, 2000, <Di>, <St>, <In>, <Pa>, <Po>, <Md>, <St>)
<Di> = (sender)
<St> = (busy)
<In> = (single)
<Pa> = (explicit)
<Po> = (immediate, indefinite)
<Md> = (main)
<St> = (pessimistic)
```

Result of capability description query.

A sample capability query obtained for a lecture session, with the professor as moderator, the video channel as resource, and its floor properties is shown above. The session identifier is $Sid = 2374$, with starting time $5:00$ pm, ending time $8:00$ pm and flat organization.

2.2.3 Consistency

The consistency of a coordination protocol refers to the integrity of state updates and the level of synchronization at all hosts in a session. Various levels of consistency on shared

resources are possible: In *strict consistency*, updates of one atomic interaction are enforced on all replicas of a shared resource (or its output) at all times. With *semi consistency*, updates of one atomic interaction are enforced on a subset of *active* sites at all times. *Loose consistency* enforces that updates on interactions are manifested at all replicas of a shared workspace at all times. In *weak consistency*, updates of several interactions are are enforced on a subset of *active* sites at regular intervals. With *null consistency* updates of interactions do not need to be consistent and are integrated between peers on demand. The decision on which consistency model to use depends on the application type, resource, and groupwork modalities. A heterogeneous consistency model, embracing hosts and applications with varying capabilities [235], must be based on a unified representation scheme for shared resources.

2.2.4 Turn-Taking

Group coordination and facilitating technologies such as floor control are closely related to the mechanisms of turn-taking among multiple parties. Turn-taking describes the patterns, by which activities of group members are sequenced [133], including speaker (or collaborator) turns, interruptions, and passing the floor. Turn-taking has previously been mostly studied in the context of conversation analysis in linguistics and psychology [191], for face-to-face meetings [16], and computer-mediated communication [160]. On the contrary, we generalize the notion of turn-taking and the floor to the context of networked multimedia systems, to devise evaluation methods and protocols for multimodal floor control in synchronous collaboration. Our objective is to build a bridge between coordination studies in traditional linguistic domains, CSCW, and computer networking.

Related Work

In linguistic terms, conversational structure is shaped by simple *turn-taking* protocols, which define the passing of speaker control ("floor handover") among multiple parties at *transition relevance points* (TRP) [191]. Conversation analysis (CA) is an established method [151]

for describing the management of conversations. In contrast to pragmatics [138], which theoretically analyzes what may have been meant with a given expression, CA observes conversational interactions based on what is said or done, providing a means to understand the implicit rules of interaction. Two problems ordinarily encountered in a basic turn-taking system described by Sacks, Schlegloff and Jefferson [191] are competition for the speaking turn and the frequent overlaps and corrections that follow. Semantic content from a previous turn may create expectancies and impact the turn-taking behavior for the next turn, for instance in the form of "adjacency pairs" such as questions and answers, which are pairs of statements belonging together. Turn-taking also depends on the valuation of the back channel signal in a turn. Larrue *et al.* [132] describe the turn-taking procedure in chaired meetings, consisting of turn-allocation from one speaker to the next speaker, and next speaker designation, where the chairperson draws the next designated speaker from a list of requests. Moderated turn-taking avoids competition for the next turn and the collisions or overlaps implied otherwise.

Computer-mediated communication and collaboration fail in various ways to match the quality of face-to-face interaction, largely due to the lack of social presence and the degradation or absence of subtle cues such as hand gestures, facial expressions, intonations, or postures. Online presence in the notion of "being logged on" is not sufficient for synchronous cooperation to establish substantial rapport among users. It has hence been suggested that nonverbal cues, such as head-turning and gaze, are relevant for floor keeping, requesting, taking or avoiding, linking groupware to the connotation of "group-awareness". Various researchers [67, 133] subscribe to the notion that session participants can maintain sufficient passive awareness through the shared workspace interface, using "free-for-all" floor control ad hoc and based on social conventions. However, very little empirical evidence indicates that visual channels add significantly to audio support in collaborative work. Previous attempts [63, 99, 133, 147] to improve collaboration awareness in synchronous collaboration, e.g., by video and audio and annotated telepointers, have been not convincing. Having more cues available may also imply that it is easier for a speaker to hold on to the floor.

Additionally, the *reciprocity rule* ("If I can see you, you can see me") does not always hold in CSCW systems.

Sellen [201] follows up on this hypothesis and compares conversational turn-taking in technology-mediated conditions with same-room and audio-only scenarios, reporting that turn-taking was unaffected even in complete absence of visual information, even though conversationalists considered visual information as most important. The fewer cues available, the greater was the reported experience in psychological distance from conversation partners. Only same-room conversations resulted in more interruptions and fewer formal floor handovers. Simultaneous starts were as likely to occur in same room conversations as with computer mediation, indicating that this problem is inherently a floor control problem. The conjecture was that better system design may be able to compensate for the shortcomings of current video conferencing systems. A similar study by O'Conaill *et al.* [163] compared video-enriched conversation over ISDN and an optical LIVE-NET with face-to-face meetings. ISDN mediation resulted in fewer backchannels for feedback, less anticipation of turn endings, and more formal interaction. Interaction over LIVE-NET was also highly formalized

Isaacs *et al.* [113] report that Grice's interaction maxims [94], which comprise quantity (as little as possible, as much as necessary), quality (truthfulness), relation (relevance), and manner (clarity), are more pertained to in computer-mediated interaction than in face-to-face meetings. A study by Watabe *et al.* [230] indicated that speaker distinction decreases rapidly with increasing group size.

McKinlay *et al.* [151] studied user conversation behavior with a text-based WYSIWIS collaboration system, comparing explicit turn-taking (ETT), using status icons signaling ready-to-talk and ready-to-listen cues, versus implicit turn-taking (ITT) in the absence of such cues. They observed that social presence itself constitutes a back channel to sustain mutual understanding. The mean turn pause length for ETT was on the average significantly shorter than with ITT, which is representative for the concept of passive collaboration awareness. Active, explicit turn-taking, through status icons and floor control is more successful because of its non-ambiguity. In addition, three policies ("computer-mediated com-

munication conditions") have been studied, free-for-all (FFA) request-and-grant (RAG), and request-and-capture (RAC). In FFA, subjects would establish their interpersonal turn management rules. In RAG, a user request was placed in a queue and served on a first-come-first-served basis, after the current turn-holder had finished and explicitly relinquished the turn by pressing a button. In RAC, any user can seize the turn anytime, independent of the current status of the active floor holder, which allows for interruptions at any moment.

McKinley also points out that in FFA, subjects contend for attention, contrasting RAC, where subjects contend for the floor. RAC promoted "wrestling" for the turn, involuntary turn giving, and immediate attempts to recapture the floor in reaction to an involuntary change of turn. Groups of various sizes were tested in a computer-mediated vs. face-to-face context, showing that RAG was the most successful turn management scheme, placing FFA second and RAC third. The study stresses that computer-mediated turn-taking, unlike speech, is not memory-less, but a semi-permanent channel, where problems associated with increase in group size can be suppressed in comparison to face-to-face meetings. According to this study, the use of signaling for turn-taking support did not detract from communication and helped to reduce pauses and overlaps in small groups. It is hypothesized that group size may be of less importance when turn management protocols permit some degree of parallel activity.

Walker [229] argues that anticipation of one's turn is essential for smooth conversational turn-taking. Detection of and reaction to turn-taking signals are deemed a high-level social skill rather than a simple stimulus-response mechanism. Turn-yielding and turn-taking signals to mediate floor apportionment are composites of various linguistic and nonverbal cues. The steps in the perception of, and reaction to a turn-yielding signal consist of an utterance stage by a speaker A, a cue inspection time V, a period of length D, in which another person B decides to speak and prepares, and a time B to activate the speech. A $200msec$ pause was identified as a criterion for a turn-triggering pause.

Activities and Causality

Our focus in previous sections has been on entities and their states in a collaborative environment. A different way of looking at interaction in CSCW systems is by describing activities and behaviors [31], in order to specify action structures and their safety and liveness conditions. We understand collaboration as a coordinated series of activities, i.e., operations on shared resources, and represent activities with a causality relation to describe the operational semantics of sharing. A turn denotes the period from the beginning to the end of an activity. Combining turn-taking with causality conditions allows characterization of collaboration with a partial precedence relation to indicate separateness or conflict freedom on a shared resource.

Let $\mathcal{A}$ be a countable set of possible activities in a session, and let $A_p \subseteq \mathcal{A}$ denote the set of possible activities. The total function

$$\alpha_R : A_p \to T \tag{2.8}$$

associates with every activity $a \in A_p$ a turn $t = \alpha_R(a)$ on resource R. We also assume a partial relation "$\leq_c$" in A_p, defining a causal order [128] on $a_1, a_2 \in A_p$:

$$a_1 \leq_c a_2 \tag{2.9}$$

This means, either that $a_1 = a_2$, or that the activity a_1 happened before activity a_2, i.e., a_2 cannot start before a_1 is completed. The triple $(A_p, \leq_c, \alpha_R)$ is called a activity structure or *task* on R, which is called *sequential*, if the ordering $\leq_c$ on A_p is linear. Two tasks $p = (A_p, \leq_c, \alpha_R)$ and $q = (A_q, \leq_c, \alpha_R)$ are isomorphic, denoted as $p \asymp q$, if and only if there exists a bijective mapping $\varrho : A_p \to A_q$ such that

$$\forall a_1 \in A_q : \alpha_p(\varrho(a_1)) = \alpha_q(a_1), \tag{2.10}$$

$$\forall a_1, a_2 \in A_q : \varrho(a_1) \leq_c \varrho(a_2) \Leftrightarrow a_1 \leq_c a_2 \tag{2.11}$$

For a given task $(A_p, \leq_c, \alpha_R)$, the relation $a \leq_c a_2$ between two activities a_1 and a_2 can be interpreted as follows:

1. Activity a_1 is causal to activity a_2;

2. Activity a_2 uses an outcome of activity a_1, i.e., a_1 cannot happen concurrently with a_2;

3. Activity a_2 starts after activity a_1 has finished;

4. There exists an activity *sequence* $a_{i,1}, a_{i,2}, \ldots a_{i,l}$ such that $a_{i,1} = a_1, a_{i,l} = a_2$, and for all $j \in [1, l-1]$, the activities $a_{i,j}$ and $a_{i,j+1}$ are either causally related by 1. - 3.

Condition 3. implies 1., but not vice versa. An activity is called *sequential*, if the causal ordering relation $\leq_c$ is linear. If a_1 and a_2 do not causally precede each other, they are called *independent* and may be carried out concurrently without floor-controlled mediation. Floors must be granted to activities that are mutually exclusive (ME). For a given task p, we define the *ME-predicate* for two activities a_1 and a_2 in turns t and t' as

$$ME(p, t, t') \equiv \forall a_1, a_2 \in A_p : \alpha_p(a_1) = t \wedge \alpha_p(a_2) = t' \Rightarrow a_2 \leq_c a_1 \vee a_1 \leq_c a_2 \qquad (2.12)$$

Observation of ME preserves the consistency of the shared workspace. Collaboration consists hence of sequences of activities aggregated in turns $t = (a_1, a_2, \ldots, a_i)_R$ on a resource R. Viewing activities as building blocks for executing tasks, causal precedence applies also to subsets of a sequence ("partial activity") and sequences of sequences ("composite activity"). Activities can be either goal-oriented, e.g., finishing by a specific deadline, or process oriented, e.g., by following a given agenda, but without goal-orientation. Outcomes of activities may be transmissions of streams (audio, video), state changes in devices (camera pan left), or read/write operations on objects in the shared graphical workspace.

Turn-Taking Model

Note that the above studies have been conducted with various simplistic chat applications and video conferencing systems in small networks, analyzing the shortcomings of CSCW to replicate face-to-face interactions. We subscribe to the view that more sophisticated, automated turn-taking strategies in collaborative multimedia systems, implemented as network support for applications, will augment social protocols, compensate for lack of social presence, and add capabilities such as playback, undo, filtering, while enabling interaction across long distances. Based on the conjecture that coordination and cooperation across computer networks are based on structuring rules similar to those in linguistic discourse, we define a turn-taking model to regulate concurrent activities on shared multimedia resources using floors. Our objective is to assess turn-taking between users, not just between users and the computer, from a network protocol perspective, adapting conversational concepts to the computational domain.

Definition 7 *A turn consists of a sequence of activities and pauses by a floor holder. A floor holder gains the floor when he or she begins accessing a resource to the exclusion of any other session member. The duration ("floor lifecycle") of a turn is delimited by the time the floor is granted to the time when the floor is relinquished, excluding propagation times to signal these changes. A group turn denotes a collaborative activity where two or more session members are active together. A floor handover occurs whenever one person or group looses the floor and another user or group gains it. A turn is called correct if the floor management governing the turn is correct.*

A turn, as a sequence $T = (a_1, a_2, \ldots, a_i)(R_j)$ of activities a on a resource R_j, must guarantee atomicity of the performed operation, and consistency at any random snapshot of the collaborative system. Its 'floor lifecycle" includes signaling and usage from onset to handoff. The partial order $(\Theta, <_H)$ defines a *collaborative history* H for a series of turns, where Θ is the set of operations executed during turns T_i in a sequence of turns

$(T_1, T_2, \ldots, T_k)$, where k is the turn count. Figure 2.10 depicts an abstraction for the flow of turn-taking messages between a local user L and a remote user R.

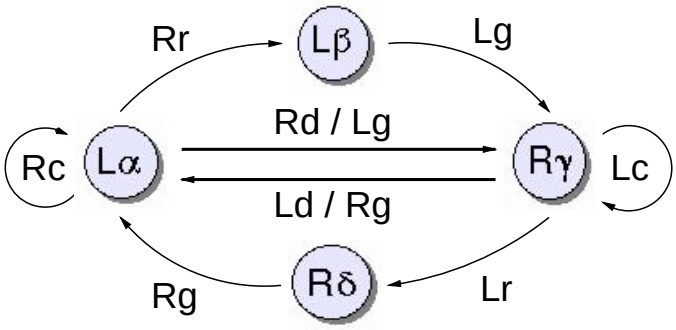

Figure 2.10: Turn-taking control and activity flow abstraction.

Turn-Taking and Floor Control

Interpreting the abstraction as a state transition diagram, the four nodes represent control or activity states of users L and R, and the transition labels represent the events or control signals triggering the state transitions. This model is a basic two-party rendition for a generic floor control mechanism without moderation and can be generalized to the n-party case with maximally $\frac{N(N-1)}{2}$ pairwise flows. Turntaking results in a $L-R-L-R-L-R...$ hand-off pattern for a particular floor, with varying floor holding times per turn. Each node may assume the role of source or receiver at the same time for multiple, concurrent interactions. The model in can be interpreted twofold, depicting the flow of coordination primitives for control, or activity states, and transitions among the two parties:

1. The set of states and events models the *control flow* in two-party interaction, with α and γ denoting "holding the floor", and β and δ denoting "concludes and relinquishes the floor". Rr denotes the event "R requests the floor", Rg represents "R gives up the floor" (alternatively, "R grants the floor"), Rc denotes "R permits 'continue' ", and Rd "R demands the floor" (symmetric for L).

2. The set of states and events models an *activity flow* between L and R, with $\alpha, \beta, \gamma, \delta \in R_a^t$ for an activity a on resource R of type t. For instance,

$$R_a^{audio} = \{talk, listen, pause, mute, replay, \ldots\} \tag{2.13}$$

The events r, g, c, d indicates state transitions to trigger the respective activities. Certain activities may be forbidden or void in a given combination, depending on resource types.

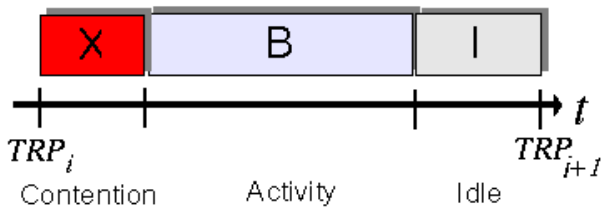

Figure 2.11: Conceptual model for turn structure.

Figure 2.11 depicts the generic floor timeline of a turn on some resource R during session time t, consisting of the contention period X, the activity or busy period B, and an idle time I. The contention period may overlap with the busy and idle periods of the previous floor holder, and models the time in which the previous holder, a moderator, or a floor allocation mechanism determine the next floor holder. The activity time is the floor lifecycle time. The idle time models the handover period, in which the next collaborator decides to request the floor and the current holder is formally holding the floor, but not using it. Turntaking relevance points (TRPs) mark floor handover moments, recognized with awareness mechanisms or social protocols, or implemented with a floor passing mechanism, e.g., using a token.

Based on the turntaking structure, we propose a generic protocol capturing the collaborative semantics of resource usage, with H denoting the current floor holder, and N as the next holder in line. The protocol is media-type independent and considers causality, multiple and codependent floors.

1. Rules applying to the first TRP of any turn:
   (a) N must obtain a floor on a shared resource before executing an operation on that resource.
   (b) If H selects N for next turn, H must relinquish the floor and N takes the floor in the follow-up TRP.
   (c) If H does not select N, then any other party may self-select, with the floor being granted to the first person according to the policy in use (default FCFS).
   (d) If H has not selected N, and no other party self-selects, then H may continue and claim the floor for the next turn.
2. Rules applying to subsequent TRPs:
   (a) Rules 1. (a) - (d) apply to determine the next floor holder.
   (b) On a turn and for any two operations p and q, if the floor for p is compatible with the floor for q, the floor for q may be acquired after the completion of p, such that q observes the effects of p.
   (c) Multiple instances of floors for the same resource R_j may only be assigned to node sets, which are disjoint, i.e., for two floor holders H_1 and H_2 and node sets N_m and N_n, $(H_1 \cup N_m) \cap (H_2 \cup N_n) = \emptyset$, $m \neq n \leq |N|$.
   (d) Codependent floors (for synchronized media such as audio and video) must be acquired and released together.

The average turn distribution H among collaborators can be measured [201] with

$$H = -\sum_i p_i log p_i \tag{2.14}$$

where p_i is the proportion of the total number of turns in a session by user i. This allows to assess the average uncertainty (or information) about who has the floor at a given time, plotted in Figure 2.12. The graph corresponds to the intuition that low-profile and highly active users are more prominent as floor-holders in a session than users that attain the floor with average frequency.

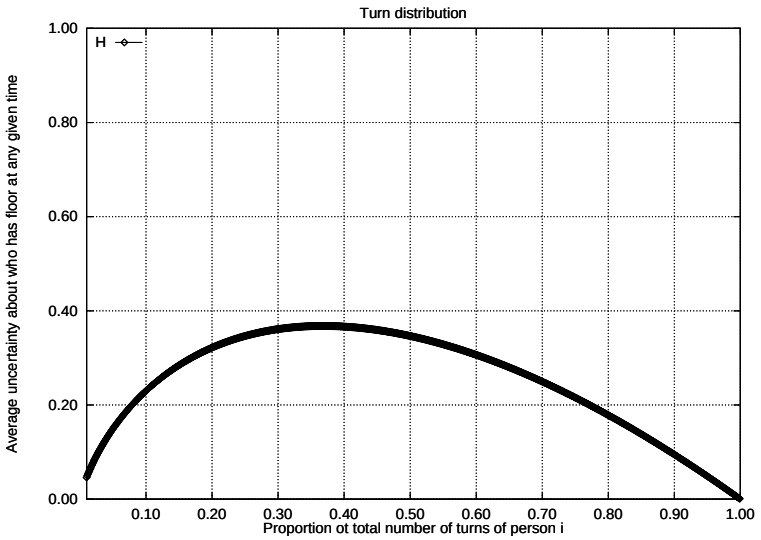

Figure 2.12: Average turn distribution.

2.3 Architecture

We discuss relevant concepts in the implementation of a group coordination architecture. Navarro *et al.* [156] identify the following requirements for CSCW systems: support for information sharing, communication, tailorability, and transparency of organization, time, views, and activities. Among the many systems built to date, we can distinguish between architectures where computers and video enrich real-world media spaces [165], such as CaveCat [27]; systems fusing virtual and face-to-face groupwork, such as the DigitalDesk [232], Clearboard [116], or the TeamWorkStation [117]; and conferencing and collaboration based on networked computers [197]. The latter being our focus, it is not clear to date where group coordination services should be deployed in the network protocol stack to fulfill these requirements, constrained by individual, group, system, and network behavior. Previous coordination protocols have been implemented at the transport layer, in the session layer as middleware component, or integrated into an application itself. The number of floors f to be tracked for a particular host is a function of the number of shared applications a, the number of resources per application r, and the granularity g, at which resource access is being controlled, $f = a \cdot r \cdot g$.

We distinguish between *collaboration-unaware*, *collaboration-aware*, and *collaboration-transparent* architectures. Unaware systems offer no intrinsic collaborative capabilities, and require external mechanisms to tap into, filter, and multiplex operations on shared resources of the session. Aware systems offer inherent collaborative functionality visible to the user and are of limited use when operated stand-alone. Bentley and Rodden [23] discern between view-level sharing on identical information with different views, and object-level sharing using replication for independent manipulation.

Collaborative systems can be *tightly* or *loosely coupled* [167]. Tight coupling presumes explicit membership control of participants and conference control enforcing a formal session style with floor control, which is for instance desirable with IP telephony. Loose coupling is based on a "light-weight" session model [76] and does not require explicit membership or conference control to synchronize media and regulate interactions. This model, implemented with the MBone conferencing tools [143], allows for anonymous, large-scale conferences, and benefits from cross-media coordination mechanisms, since users want to see their activities manifested in coherent responses by all tools. Mechanisms to bridge the gap, such as a moderated blackboard for posting questions and stimulate interaction have been built for the MBone [146].

2.3.1 Paradigms

At the highest level of abstraction, a prototypical collaboration environment contains five components:

- An Orchestration Manager for conference or session setup, membership tracking (invitation, joining, leaving) and teardown (OM).
- A Coordination Manager for exercising group coordination services such as floor control on communication channels and network resources in the shared workspace (CM).
- A Workspace or Window Manager, which integrates the local, private workspace, and the public, shared workspace, where the latter can be rendered in local, independent

views, or in synchronization with remote hosts (WM). The underlying information base for WM may be local or obtain data from remote servers.

- One or more Shared Resources, such as an application or device (SR).
- A network interface linking the above components.

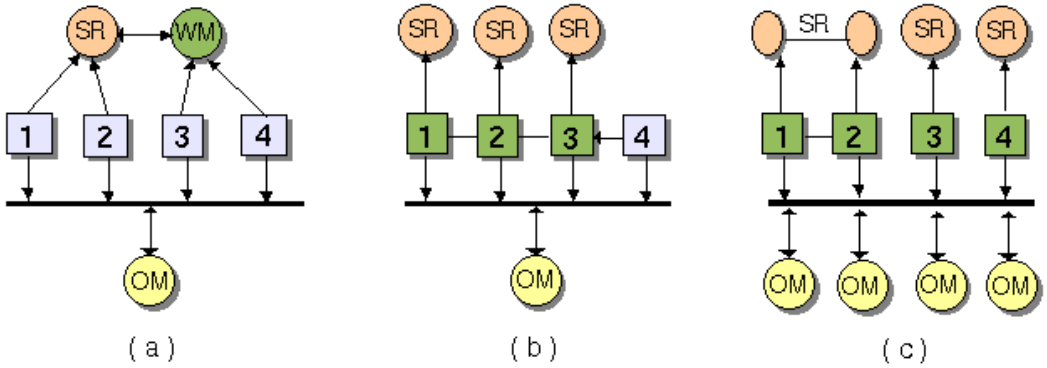

Figure 2.13: Coordination configurations: (a) Centralized; (b) Hybrid; (c) Distributed.

The CM may exercise control by intercepting messages sent between WM and SR, or by processing control requests from the various nodes. For implementing group coordination, we have a choice of centralized or distributed coordination management, or hybrids of both. OM and CM may be centralized, a hybrid of centralization and distribution, or purely distributed, as depicted as in Figure 2.13. Darker-shaded hosts, in contrast to light-shaded hosts, have an integrated WM and have the option of independent views. In *central sequencing*, all changes to shared resources are funneled through a single host ("floor server"), with the benefit of predictable delays, simple implementation logic, and consistency. Pendergast [174] distinguishes between three types of central processing: terminal linking, centralized data, and distributed data. Terminal linking yields strict WYSIWIS, centralized data lets hosts obtain data through a central server, but process them locally, and in distributed data all shared information is resident at the hosts' local storage and a central node sequences all operations. The first and second solution offer inherent consistency, but incur high network traffic. The third solution implies partial or full replication

of data in a federation of servers or hosts, and is the predominant approach for collaborative systems, with the advantages of low network traffic and superior response, but with difficulty in maintaining consistency.

In Figure 2.13 (a), all hosts sequence operations through a central OM, with hosts 1 and 2 accessing a resource directly with WYSIWIS, whereas hosts 3 and 4 use relaxed WYSIWIS by employing a WM for independent views. In the hybrid approach (b), host 4 communicates solely through host 3 to access SR, whereby hosts $1-3$ maintain their own replica of SR and sequence exclusively through a central OM. In the distributed case (c), hosts 1 and 2 possess partial information on a resource SR and are linked to indicate exchange of information to mutually complete their picture of the resource. Hosts 3 and 4 access SR as independent objects, which exemplifies the special case investigated also by Pendergast [174], where no special coordination mechanism is needed to correlate state information, because resources can be manipulated synchronously without violating mutual consistency constraints. In all cases, we assume for simplification that OM contains CM, e.g., when a session manager contains or interfaces to a coordination agent linked at run-time with other coordination agents.

Most existing conferencing and collaboration systems can be subsumed under one of these models. In a centralized scheme, all hosts tap into the same application and use a single CM, with the shortcoming that hosts cannot be decoupled and depend on CM and the application server as a performance bottleneck under high load and point of failure. Centralized coordination incurs no integration cost. In a hybrid scheme, each host has its own replicated information base coordinated through a central coordination protocol. Hosts from the same local network partaking in a wide-area session communicate with a local conference server through a local bus architecture, which attaches to another conference server on the conference backbone. In a distributed architecture, CM, WM, and SR are all host-individual, and coordination is peer-based using a conference bus, with the benefits of flexibility in session organization, load-balancing, scalability, and fault-tolerance. Several systems with floor control representing these paradigms will be discussed in Chapter 3.

2.3.2 Design Issues

We postulate henceforth a coordination architecture based on the following requirements: simplicity of implementation and maintenance; scalability in the number of users and hosts [196]; security with regard to the exchange of coordination information and data; extensibility for new resources and session models; efficiency in coordination, concerning low latency and protocol state overhead; reliability with regard to failures of hosts, resources, and network elements; persistence of coordination information at hosts despite the ephemeral nature of floors; and interoperability between heterogeneous platforms.

The maintenance and distribution of coordination state information should be *partially or fully distributed* to avoid, for instance, the problem that a single floor server becomes a bottleneck and single point of failure for a session. Distributed session orchestration also allows for continuation within partitions of a split session, in case that one or more links between hosts fail. Session information should be *logged* at several hosts for the case that disconnected sites resurface and attempt to rejoin a session, retrieving updates on the collective session state from one of the active session partitions. Shared workspaces should be *asymmetric*, that is, each host is allowed to maintain its own individual view on the shared resources, but may synchronize itself a WYSIWIS state, if required. Session updates, rather than flooding the network, should be sent in increments between hosts, encoding the difference in previous and current work states.

A *hierarchical* host organization and representation of the workspace allows for more efficient naming, addressing, and state keeping of large sessions, and inheritance of globally valid session attributes. A hierarchy models face-to-face meetings better, because users may want to temporarily step out of a larger session to conduct a side-chat without having to define a new multicast group. Sessions should allow for various orchestration models, from tight control to loose control, to cater to the need of registered, formal meetings and informal meetings. Floor control can be applied in both orchestration styles and should take into account the demanded Quality-of-Service in relation to the available service quality feasible with current host and network conditions.

Despite a considerable body of work on floor control, concurrency control, or access control in collaborative systems, there is no consensus or methodology on the implementation and deployment of floor control services. Ideally, coordination services would be based a common structuring principle, such as a common dissemination and control geometry. Most available literature treats floor control as a black box service without providing details on operation or implementation. The MCP protocol [238] uses a conversational abstraction for floor control on top of the transport-layer multicast protocol XTP [214]. The goal is to achieve temporal and causal synchronization among concurrent flows between multiple hosts using a token-passing mechanism. On the other hand, floor control has also been proposed as a bridging component in call management [6], or as a session layer component [198], as for instance in the conference control protocol CCCP [104].

Floor control may be centralized, while session control is distributed, or vice versa. In contrast to this modular solution stands the monolithic approach of integrating floor control into the application itself, which is the prevalent solution for the majority of collaborative systems, such as the collaborative visualization system CSPRAY [169]. We subscribe to the view that the provision of coordination services such as floor control as a middleware component at the session layer achieves better flexibility and re-usability in comparison with transport layer or application layer deployment. We refer to this modular architecture of deploying floor control as a sibling component with session control as "Floor Assignment in Collaborative Environments" (FACE). In FACE, each host runs a floor daemon enforcing control over resources in the shared workspace and communicating with remote hosts on the status of their floors and resources. Figure 2.14 shows the layering of host-based services in a networked multimedia system, with coordination services located between the application layer and the media services and network support layer. Synchronization refers to playback synchronization of media streams [141] and is attributed to the media layer, in contrast to activity synchronization between users [9]. Our conceptual model of a group coordination architecture includes security and activity synchronization services, which cannot be addressed within the scope of this thesis.

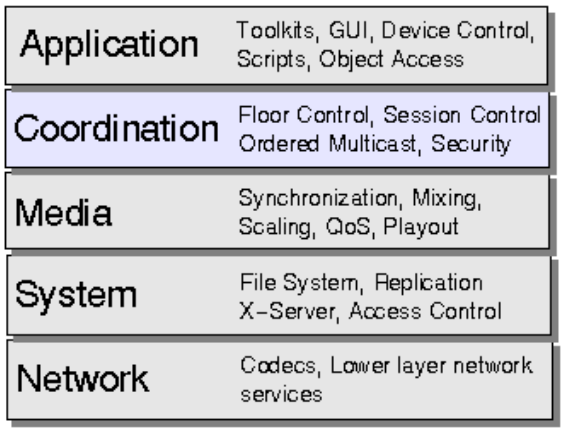

Figure 2.14: Group coordination architecture.

While floor control handles coordination, session control keeps track of membership, start-up and tear-down of the communication paths, and the use of compatibility issues with particular media. Session control complements coordination mechanisms such as floor control and performs membership management [178], directory services [102], announcement [101], invitation [103], and teardown. Integrated session and floor control services have been referred to as conference control [104]. Typical membership functions fall into four categories: registration calls such as invite, initiate, join, etc.; modification of status and authorization; polling of information about the remote or local status; and termination, including functions such as leave, withdrawal, or expel.

Floor control is often associated with tightly-coupled sessions, but sessions without predefined membership such as panel discussions may need floor coordination software as well. Session control mediates between upper application layers, and relays requests down to end-to-end services. It comprises the following tasks: initiation, pause, resume, and stop of sessions. These are characterized by their purpose and a set of resources. Session participants are validated and tracked via a directory based upon their group membership. Basic membership support includes creation, joining, withdrawing, inviting, excluding etc. by single members or whole groups. A session control protocol also needs to account for concurrent membership in parallel sessions, switching, overlap, recursive establishment of subgroups and collaboration across session boundaries. The group coordination mechanisms

put forward in this thesis assume communication based on message passing in a packet-switched network.

2.3.3 Aggregation

We conclude this chapter by making the case for aggregated processing of coordination information in a hierarchical host topology. This principle will be of use for the design of efficient multicast-based floor control and message ordering protocols. Group coordination relates to multicast routing and reliable multicasting [162], because control messages must be routed among hosts in the control tree built for managing session interactions. Failed control directives must be retransmitted, similar to packet loss recovery in reliable multicast. Even though more recent collaborative applications use the IP-multicast model for dissemination of streams, this model alone seems not sufficiently powerful for the spectrum of distributed multimedia applications.

A multicast tree is either a shortest path tree, which is a directed tree by one source reaching all members of the multicast group, or a shared tree, which is constructed for a group and shared by all sources. The multicast delivery tree is constantly pruned or extended by a multicast routing protocol such as DVMRP, CBT or PIM-SM (cf. [112] for an overview) reflecting the current subscription state to multicast groups and the presence of adjunct network resources. Multicast group membership is provided by a membership protocol such as IGMP [74]. Source trees are suited for a scenario where one source incites a long-lived transmission to other session members, and no further individual source trees in the session must be built. For multimedia collaboration with frequently switching sources, a shared tree is a more effective solution, with the shortcoming that delivery paths in the shared tree may be suboptimal and affect the latency of continuous media.

Consider a scenario where three hosts from three different multicast groups MG1 - MG3 must coordinate access to a shared resource they are contending for. Figure 2.15 depicts a snapshot of the protocol operation in of telecollaborative session across a ternary tree displayed to the right.

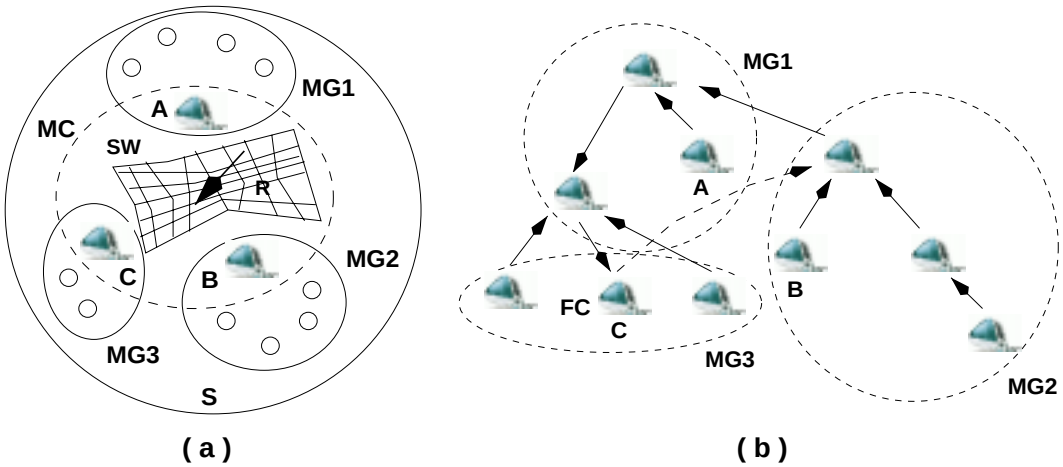

Figure 2.15: Snapshot of group coordination among multicast groups MG1 - MG3 and corresponding dissemination tree.

The resource R under contention is a data grid with a shared telepointer, which can be rendered differently depending on model assumptions and parameters given. For example, it may display wind velocities, or a three-dimensional temperature field. Three multicast groups of researchers, MG1 - MG3, work on this shared space, whose group affiliation indicates different interest in the data. Among them, users A, B, and C from each group form a multicast coterie MC, that is a subgroup, for example with a special interest for wind data. With the standard IP-multicast model any wind data renditions made by user C will not only be visible to coterie members A and B, but to the members of all groups.

Not only may receivers not be interested in such content, but transfer of rendition data may also waste unnecessary network and host resources. One possibility for the researchers in MC is to form an extra multicast group, but if many such intermediate results need to be created, a more elegant and transparent method must be used to exchange interim computation results. It is hence desirable to allow transmissions to subgroups of multicast groups and for data to be subcast on a per-packet basis. For such highly interactive group work, the per-source tree model would require hosts to join a new tree per turn, and subsequently tear down the temporary multicast tree, which is impractical.

In an alternative model, a single shared tree is constructed in the beginning of a session and hosts join the session by being added into the tree. When a host become floor holder,

it transmits its data to its children, if the target hosts are located in its subtree, or to its parent host, if the target is located elsewhere in the tree. Each transmission involves therefore only as many hosts as the branching factor of the tree indicates. In case of stale links or failed hosts, many heuristics have been proposed on how to reconstruct and optimize shared trees. This motivates a refined intra-group addressability service to allow selective multicast of control information and data to subgroups of large groups on a per-turn or per-packet basis. The IP-multicast model lacks addressing information, by which elements of multicast groups could confer with each other, without affecting the session as a whole. A floor holding host as a sender in a collaborative session could hence only address an entire group. We discuss a novel approach that integrates results from work on extended multicast services [136] with group coordination.

Attaching positional labels to nodes in a D-ary tree implies an additional storage cost of $log_2 D$ bits per level in a positional tree of N receivers and height $log_D N$, i.e., lgN bits are needed. Using 32-bit labels for designating sources and targets in message headers, up to 2^{32} hosts can hence be accommodated. Prefix comparison is cheaper for nodes close to the root due to shorter labels. The message cost to handle a CP is

$$C_{CP} = c_{req} + c_{resp} + c_{upd} \tag{2.15}$$

consisting of the cost to send a request to a control node, receive a response, and multicast an update on the new state. We compare the delay in a unicast, multicast, and aggregated multicast communication model under full load (each node sends a CP), assuming that the host processing cost for request, response and update packets is equal and normalized. The average path length between nodes is assumed to be the same for all models. λ represents the individual processing, packetization and transmission delay for each type of packet.

In unicast, the coordination delay incurs $(N-1)$ requests, replies from control nodes, and updates, where N is the current session size, i.e., $CD_{uc} = 3(N-1)\lambda$.

In multicast, $(N-1)$ nodes send requests, and the control node multicasts one reply and one update back to the session, thus $CD_{mc} = (N+1)\lambda$.

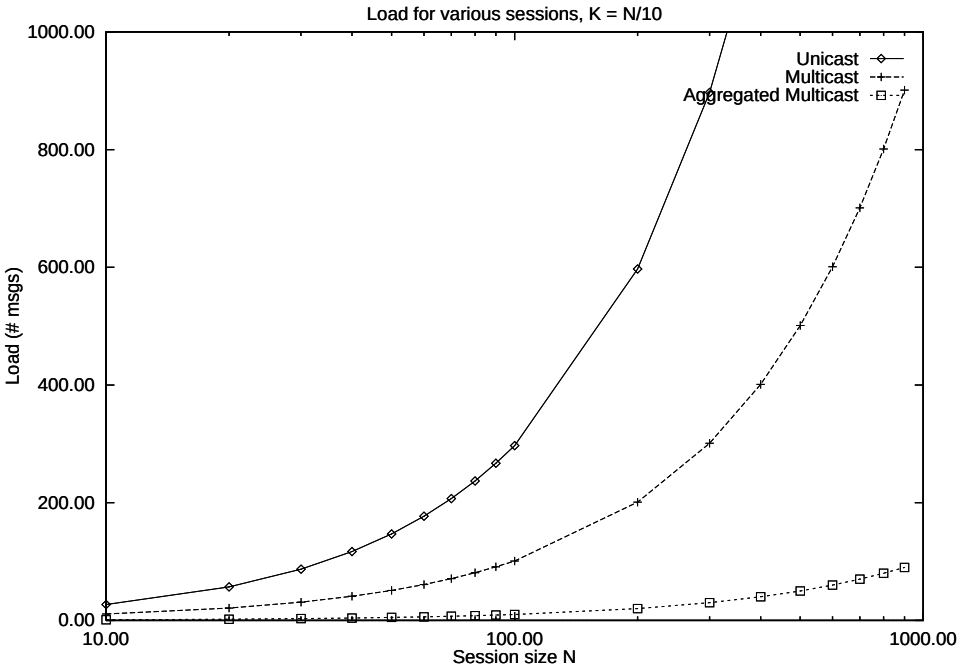

Figure 2.16: Message cost for coordination with unicast, multicast and aggregated multicast.

In aggregated multicast, CPs are handled within multicast groups and only the root of a group forwards a composite request to its parent, or responds to group-local requests, if it holds the information locally. With K groups we have on the average $G = \lceil \frac{N}{K} \rceil$ members per group, and per group there are G requests inside a group, K aggregated requests sent to a control node from all groups, and one multicast response and update, hence $CD_{amc} = ((G-1) + (K-1) + 2)\lambda = (G+K)\lambda$.

Figure 2.16 shows the average cost to coordinate hosts in sessions up to size $N = 1000$, clustered into $K = N/10$ groups, and with normalized transmission delay λ. It elicits the benefits of aggregated multicast coordination.

2.4 Discussion

Dynamic sharing of online work is a new paradigm with consequences for communication and data processing that will become more apparent for the years to come. Applications will increasingly offer collaborative services, as can be seen by the current trend to make web-

browsers more interactive. Group coordination is a promising methodology for improving cooperative behavior between users and agents to utilize distributed system components, complementing group membership and group dissemination services. We discussed a formal model of coordination, including relevant entities and their parameterization, a causality-driven activity and turn-taking abstraction, and the design choices in implementing group coordination services.

In the following chapter, we discuss floor control as a central component in group coordination to mitigate contention in simultaneous creation, access and manipulation of resources in networked multimedia systems. We will use the idea of aggregation to integrate floor control with tree-based multicasting.

Chapter 3
Floor Control

We understand floor control as a technology to implement group coordination in collaborative environments, mitigating race conditions in concurrent usage of shared resources. Distributed activities using continuous media and discrete resources are orchestrated with a floor control mechanism. A prime example of a floor control problem is turn-taking on the audio channel to allow for smooth multiparty conversations. If everybody would be allowed to take the speaker floor, packets from different sites collide and messages get scrambled. Backoff and retrying, chaired control or mediation through other communication channels are possible, but inefficient solutions to resolve such conflicts. Regulation of turn-taking in electronic meetings is problematic, in particular for larger groups or communication across long distances due to the delay in signaling floor capture. A floor control protocol ensures that turn management is safe and live by arbitrating resource usage with floors.

Controlling the floor helps users to regulate and synchronize collaboration, gain more flexibility, engagement, and decisiveness on who is in charge in a session, promotes fair turn-taking in resource access among session members, defines various degrees of formality for meeting conduct, and promotes group awareness. Bandwidth may be used more efficiently, for instance, when video streams for passive participants are toned down and only the current speaker's video content is transmitted at highest quality, establishing a user-originating mechanism to throttle data flows in session links.

Floor control is a complex subject because non-separable user, end-host, and network issues need to be taken into account in design and implementation, considering scalability,

distributed state management, consistency and reliability. Previous work on floor control has either focused solely on user-interface issues, or tackled system-level issues based on a unicast transmission model and limited choice of media. Existing descriptions of floor control protocols lack detail and analysis. Our goal is to develop a methodology for floor control which allows implementors and users of collaborative systems to make informed choices on how to use coordination in group-enabled software. We comprehensively report on previous work, summarize important concepts of floor control, and provide a taxonomy and performance analysis for the various paradigms in existence. Two novel protocols, for direct-link, unicast networks, and for hierarchical floor control are presented, where we extend floor control mechanisms to a multicast dissemination model.

3.1 Related Work

The core ideas of providing floor control in computer-supported cooperative work stem from work on turn-taking, as discussed in Chapter 2. A very basic rendition of floor control is integrated in text-based UNIX conferencing tools such as `talk` for a chat session between two parties, `ytalk` for 3-way sessions, and `confer` or `joinconf` for small multiparty chat sessions. Turns are marked with the cursor alternating between the local window and remote windows to mark the active writer. In `confer`, chat partners join based on the invitation by the session initiator, and the name of the current floor holder is displayed on all screens in brackets. The floor can be claimed by pressing the Enter-key and relinquished by entering a blank line. Usage of these tools is limited to UNIX systems, but similar tools are available on any platform.

The NLS/AUGMENT system [71] was an early text processing system in a networked, multi-user environment, supporting multiparty collaboration in the form of file-sharing, email, and "televiewing". A gavel passing mechanism was conceived to "pass control back and forth between workers", using multiple windows and replication of content from a users terminal to other users' screens, and for "subsequent entry and departure of other conference participants."

Forgie [78] discusses various concepts of activity coordination in packet-switched voice-conferencing systems, inspiring Aguilar *et al.* to propose a distributed voice-activated collision-sensing algorithm for the EMCE teleconferencing system [5, 82, 83]. EMCE supports a free-for-all floor policy in a local area network assuming ample bandwidth and low delay, allowing for backchannel floors. A derivative of this idea, an *activity-sensing* floor acquisition strategy, has been proposed by Garcia-Luna *et al.* [84, 175] for LAN-based telecollaboration, where sites backoff from claiming the floor when they perceive remote activity. Based on this design, Craighill *et al.* [41, 42] implemented the task-activated floor control algorithm COMET in the collaborative engineering system CECED, where X-server events are monitored and intercepted, when handling shared applications resources. With this mechanism, non-collaborative applications have to be altered only minimally to equip them transparently with groupwork capabilities. The paper also put forward ideas on control with finer granularity and the combination of policies, role-based authorizations, and multicast with floor control.

The XEROX COLAB [211] system was an experimental meeting room designed to support collaborative processes in face-to-face meetings with collective sketching, presentation planning and proposal planning among two to six persons. It was equipped with a LAN of workstations and a large touch-sensitive screen, and a coordinated multiuser interface provides consistent presentation of images of shared information to all participants. Such totally synchronized views are referred to as the "What You See Is What I See" (WYSIWIS) metaphor. A *relaxed* version of WYSIWIS allows users to obtain individual perspectives on the same content, differentiating between public and private windows. Four versions of concurrency control on the COLAB database were considered: a centralized model; a replicated model, where all users maintain a local cache of shared records and must attain ownership to make changes; a cooperative model, where participants use verbal "voice locks" to prevent inconsistencies in broadcasting updates of their local records without synchronization; and "roving locks" to create a working set of locks on shared data. Due to inherent delays in all versions in the propagation and reception of lock updates, a dependency-detection model

using timestamps was considered as well, which implements optimistic concurrency control and depends on semantic knowledge of resource dependencies to recreate consistency.

Greif and Sarin discuss floor control for text and graphics-based scheduling applications [93], distinguishing between automatic and manual floor-passing and proposing a reservation-based floor control scheme and the use of multicast. The importance of a smooth conference phasing out is stressed, because typically not all participants leave a conference at the same time. Abdel-Wahab [1] discusses a centralized drawing tools with floor control implemented as a chalk passing mechanism between a session server and conferees. Lantz' V-Conf system [131] proposed floor control to interface with a conference agent mediating I/O between shared applications. A conference front-end (user interface and invocation of shared applications), conference agent (mediating I/O between shared applications) and a conference manager (floor control and other synchronization) are introduced as three process types. The system lacks support of voice and long-haul networks and its floor control mechanism can lead to visual inconsistencies between connected sites.

The Rendezvous architecture and language [172] were conceived to support in the implementation of applications in which multiple parties can manipulate shared graphics objects at the same time, such as a tool for the collaborative design of user interfaces. The authors argue that the centralized "abstraction-link-view" architecture of Rendezvous enables simplified concurrency control and consistency, and keeps synchronization problems among widely distributed users manageable. Enforced turn-taking between users is considered as a basic form of automated floor control, consisting of a mechanism to select the next floor-holder and the ability to enforce control.

Crowley *et al.* [43] discuss the MMConf architecture, which uses floor control to orchestrate telepointer usage in a replicated whiteboard application. Sequence numbers order distributed activities, which are logged and can be replayed in the case of failure. Each application has a floor manager, which communicates with other managers about floor passing. Applications can refuse to relinquish the floor, and floors in transit may trigger redundant retransmissions of requests, which may result in deadlock and unfairness.

Various other experimental conferencing environments have been developed offering a notion of floor control, such as CANTATA [36], MERMAID [230], RAPPORT [6], MONET [210], the fault-tolerant NEBULA system [158], the Joint Viewing and Tele-Operation System (JVTOS) [48] for shared work with telepointers, BERKOM [11] (which offers collaboration services with choice of different service policies), HIGHEND [168] for aerodynamics visualization, ARGO [81], and SHASTRA [14]. The collaborative CAD and telemedicine system SHASTRA employs a two-tier floor control mechanism based on token-passing, which supports access, browse, modify, copy and grant permissions, and resolves contention on hot-spots with a first-come-first-served policy. Distributed floor control has been implemented in MERMAID, where duplicated computations and consistent replication may be costly, and both implementation and run-time monitoring of a floor control system can become very complex. The "UnOfficial Yellow Pages of CSCW" [144] list various systems featuring floor control, but the majority lack scalability, wide-area scope, and multimedia capabilities.

Meta-applications to create collaborative systems, such as GROUPKIT [188] or COSARA [220] assist in rapid-prototyping of graphics-oriented conversational applications. GROUPKIT features a replicated run-time architecture handling distributed processes, overlays for integrating groupware elements such as telepointers, and an interface to adapt to the communication needs of a group-aware application. GROUPKIT lacks multimedia capabilities and floor control is regarded a user interface concurrency control issue exemplified with telepointers [91].

The early MBONE tools [72] VAT and RAT for audio conferencing and VIC [150] for video conferencing were the first efforts to deploy real-time multimedia collaboration services at Internet scope. The tools are cross-coupled and feature coordinated start, suspension and termination, and automatic source selection based on voice-activation. Listening users are asked to mute their microphones for reasons of echo cancellation and bandwidth conservation. The QUESTIONBOARD [146] supplements manual floor control with a moderated black-board mechanism, usable for larger seminars, where users to post questions and stimulate interaction. Recent research on MBONE tools [108, 149] and by the MMUSIC division

of the Internet Engineering Task Force (IETF) focuses on protocol support for light-weight conferences [199] that are pervasive on the MBone. The Conference Channel Control Protocol (CCCP) is designed to scale from tightly coupled groups to large scale sessions, and conceptualizes floor control as a modular unit interacting with session control across a conference bus, but the CCCP specification lacks details on the floor control protocol. The Simple Conference Control Protocol (SCCP) [29] outlines a token-based floor control mechanism allowing for shared, concurrent floors in a centralized architecture, however, this and other efforts [167] are work in progress and lack detailed specification.

The Jets system [206] taps into the current need to meet on the web with real-time interactive meeting services, and features a web-centric Java-based synchronous collaboration environment. Floor control is implemented as a locking mechanisms to prevent activity collisions in the shared workspace. In the ITU T.120 [224] standard for real-time and audiographics conferencing, designed for conferencing, document sharing and whiteboard activities in circuit-switched networks, centralized multipoint conferencing units (MCU) are used to "reflect" media streams between sites and select the current speaker.

CSpray [169] is a collaboration-aware visualization system implemented within the UCSC Reinas project [189]. It is designed for small groups to jointly explore real-time data from a network of environmental sensors for geoscientific research. The shared workspace consists of "spray can widgets" used to render data from various spatiotemporal views. Workspaces are asymmetric, i.e., each user can maintain an individual local view on the common renderings in the workspace. Only updates between different data set renditions are broadcast via UDP to participating sites in a difference scheme and in compressed format for the sake of efficiency.

Visualization data can be fed into CSpray at run-time from a real-time database and maintained as a partial replica independently by each host. The floor for a spray can is obtained by clicking on the spray can icon and specifying a spray mode. Multiple cans are allowed on the shared visualization space and are distinguished by name tags of the active users. CSpray has been demonstrated in live three-way sessions during the Supercomputing

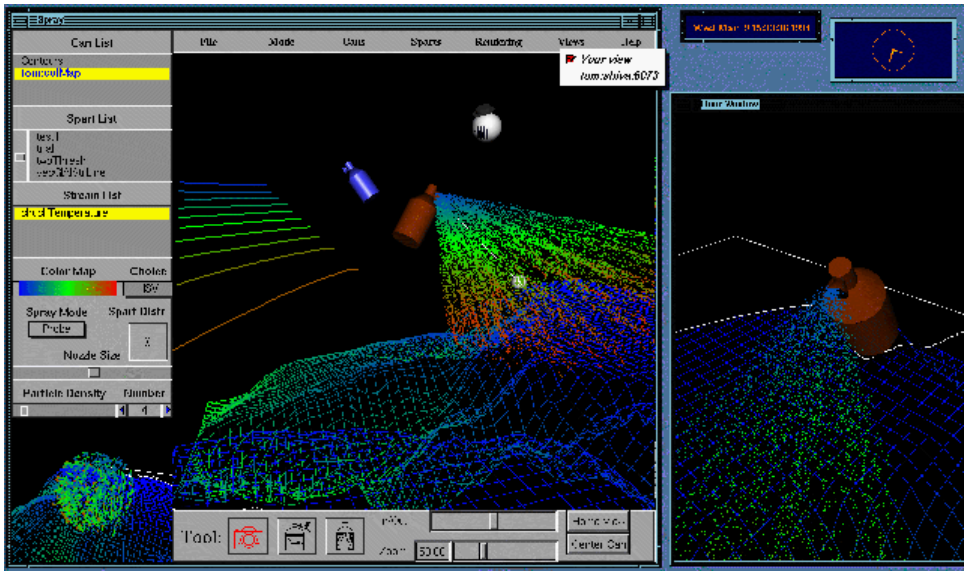

Figure 3.1: Snapshot of CSpray - collaborative visualization of remote sensing data. The spray can icon is used by a user to grab the floor for a specific rendering mode (from [169]).

conference in San Diego in December 1995 using $FORE^{TM}$ ATM backbone switches. A snapshot of a two-party collaboration is shown in Figure 3.1.

Figure 3.2 shows a screenshot of the web-based interface to CCAM, implemented as a test application for floor control on remotely controllable instruments, in our case a web-attached camera. The camera served as an end-device for demonstrating video streaming among wireless Internet gateways. In contrast to the many webcameras available to date, the idea behind CCAM was to establish more organized patterns of multiuser access to the camera, using a combination of C code, Perl, Java and CGI scripting. CCAM has been used in a GLOMO project demonstration within the WINGS testbed since 1997, controlling the camera live from multiple hosts on the Eastcoast and Westcoast.

Methods to mediate resource contention at the user interface level have been discussed for example by Ellis and Gibbs [66]. Greenberg [90] compares six floor-control protocols from earlier work, not differentiating between policy and mechanism: a free-floor scheme, a preemptive scheme, an explicit release model, a round-robin scheme, a central moderator, and pause detection, all selectable by user choice in the personalizable SHARE application. Agrawal [4] discuss the management of concurrent activities in database-centric

Figure 3.2: Snapshot of CCAM - collaborative camera control through the World Wide Web. Clicking on elements in the navigation rose grants a user the floor, when available, for the duration of the camera movement.

collaborative environments. A conceptual integration of floor control within intelligent agent architectures has been proposed by Edmonds *et al.* [65]. Yavatkar [238] described a token-based floor control mechanism in the MCP protocol framework, which implements a conversational abstraction for synchronized exchange of multimedia flows on top of the reliable multicast protocol XTP. Hilt and Geyer [107] discuss floor control requirements for distance learning environments and define a collaborative services model based on work

by Dommel and Garcia-Luna [54]. Trehel [221] analyzes floor control as a request synchronization problem for a distance learning session, taking into account the order of requests and the role hierarchy between teacher and students. Guyennet *et al.* [100] present a consistency management protocol for CSCW applications with shared memory, proposing a token-based mechanism to structure the sharing process. Kausar [124] presents an overview on the marriage of floor control services with reliable multicast.

3.2 Characteristics

In this section we delineate floor control from other control paradigms to manage distributed concurrent resource access, discuss control mechanisms vs. policies, Quality-of-Service adaptation observing control state information, and consider user interface issues.

3.2.1 Related Control Paradigms

Floor control has counterparts in concurrency control used in database system [33], message-passing mutual exclusion schemes [209] used in operating systems and distributed systems, and access and version control such as UNIX `rcs` used for file management in networked operating systems. Note that floor control mechanisms solving the multiaccess problem in groupware are occasionally referred to as concurrency control [66, 91].

Traditional database concurrency control resolves update conflicts on discrete database records that cannot be shared directly or in real-time among users, using a transaction model to serialize and carry through, or rollback, operations observing atomicity, consistency, isolation, and durability (ACID) properties. A large number of optimizations and variations on transaction management exist, with regard to locking mechanisms, multi-level updates, consistency management, and observation of domain semantics. Barghouti [20] presents a survey on advanced concurrency control methods designed for long-lived transactions and increased user control for collaborative CAD or CASE systems.

Mutual exclusion [129] solves contention problems in resource allocation, in a manner opposite to cooperation problems solved by producer-consumer schemes. Processes con-

tending for the same resource must be synchronized by semaphors, monitors or message passing to prevent two processes executing simultaneously in the critical section. However, in contention, a process must be able to make unlimited progress when others fail, while in cooperation the progress of one process depends upon the progress of another.

Access control as protection management [130] for distributed file access is tailored to non-interactive storage and sharing of files. Shen [202] proposed permission hierarchies and finer lock granularities as extensions for collaborative work, but such refined access control is limited to discrete files. Similarly, version control uses a differencing scheme and locking technique for concurrent editing, coherency maintenance and economic storage, placing a temporary lock on a file edited by a user and logging the different branches of a file's history. It has been applied for example in a hypertext-based notecards system [222], or for groupwork in a database centered context [123].

On the other hand, floor control supports interactions rather than transactions, and targets cooperation problems. It mediates direct and open-ended computer-based interaction among two or more users using continuous media or accessing discrete data objects with variable granularity. Such multimedia resources are not subject to an underlying ACID database or file system transaction semantics. Activities can be carried out pessimistically, or optimistically without the possibility to undo [176] operations. Control messages are passed between hosts and floor servers to implement ephemeral, distributed resource access. A floor-controlled access semantics includes `transmit, mute, replay, modify` and other operations on continous media, contrasting the classic notion of read-write and write-write conflicts, as they must be mitigated for databases with a predictable access semantics. Operations on multimedia data may be not commutative and are not serializable as in concurrency control, since worksteps must be carried out interleaved. Floor-mitigated access may be long-termed and floor scope can be aligned with a session, application, its components or granules. In analogy to mutual exclusion, a floor control algorithm must be capable of allowing multiple parties to gain access to the same resource, if desired, and tolerate node or resource failures in a session without loosing operational integrity.

3.2.2 Control Mechanisms and Policies

Floor control protocols can be dissected into the *mechanism* handling control message dissemination, and the user-oriented *policy* [30, 43, 203], which determines the reordering, preemption, and scheduling rules on how floors are logically assigned. A separation of these concepts permits adoption of new policies without altering the mechanisms used. A mechanism may imply a default policy, and not all policies can be instantiated for a given mechanism. Mechanisms and policies impact responsiveness, fairness, and resilience of floor control. Users agree on the floor either by signaling each other with signs or words, or interpret the absence of signals and activities of partners as permissions to acquire the floor. Accordingly, floor control mechanisms are either based on the explicit exchange of data or control tokens signifying floor state changes in order to seize control over a resource, or use implicit assertions in local variables at each host to assume permissions to seize a floor.

Mechanisms

We distinguish between the following mechanisms to handle the low-level processes to generate and stream requests among users: *Incoordination* permits users to negotiate access patterns in a free-for-all policy, where no cues or tokens are provided by the collaborative system to help users coordinate their activities. Social rules may result in order or anarchy. *Activity sensing* forces a host to give priority to activities perceived at remote sites, assigning the floor to the sender and blocking locally originating data. If no remote activity is being perceived, the local site assumes the floor, allowing local activity. Contention for the floor is resolved through randomness of floor capture events, i.e., if two sites claim the floor at the same time, both must stop requesting, back off and restart their request after a random waiting period. This paradigm is based on channel access mechanisms such as Carrier-Sense-Multiple-Access [25]. The drawback of this strategy is that hosts with fast connections are likely to acquire the floor more quickly and often and may block resource access. In contrast to schemes using explicit floor tokens, activity sensing is solely based

on contention and has no factual notion of floor control, because a host never attains the exclusive right to be the only host accessing a resource.

In *token passing*, floors are explicitly represented and acquired with a floor token, which is passed among session members similar to "gavel-passing" in face-to-face meetings. Contention can be resolved with token passing using the data channel to send control signals to coordinate activities. The order of passing may be determined by the actual physical network topology, e.g., a ring topology, the logical session topology and a predefined schedule such as round-robin, or a request-reply dialogue among session participants. In *token asking*, or token fetching, the floor token is not automatically passed within a session, but must be acquired from and returned to a coordinator node. In *floor polling*, a coordinator node offers the floor token to session members in a sequence reflecting a chosen floor policy. Explicit mechanisms use timestamps or sequence numbers to mark the request or grant time for a floor, defining an order in which requests are satisfied. *Two-phase locking* establishes floors as locks on shared documents in a growing phase and releases them after activity completion in a shrinking phase, which prohibits acquisition of new locks. Many variations on this locking mechanism can be borrowed from concurrency control [24]. In *time-slot allocation*, turn time is divided into equal time slots and users attempting to gain the floor must do so in one of these time-slots. An allocation of time-slots for floor acquisition in a round-robin fashion is equivalent to statistical time-division multiplexing. The drawback of this mechanism is that open user interaction can generally not be modeled with discrete time-slotting.

In *blocking*, system processes representing a collaborative work thread are prevented from progressing according to a session-wide floor control state, which is explicitly signaled among sites. In *reservation*, hosts may specify resource access times and quality in a allocation period, and are granted resources according to order of arrival and availability. Reservation requires a separate control channel to reserve resources apart from the data channel and is not suited for spontaneous interaction, where resources must be allocated on the fly. *Versioning* is an optimistic scheme based on file locks, used for concurrent document

access and archival, but not applicable for continuous media. *Dependency detection* permits concurrent document modifications and recreates consistency based on given underlying data semantics [211], but is also not usable for real-time media. Users may also proceed with modifications on replicated resources and subsequently *vote* on the adaptation of changes, which is also limited to static documents.

Policies

Floor *policies* determine user access based on criteria such as the origination or arrival time of a request, the desired Quality-of-Service, the average waiting time in a queue, the order by which users have previously been serviced, a priority value, or a method to manipulate resources within the user interface. Boyd [30] argues that policies depend on the degree of interaction required, i.e., automatic vs. manual input in control, the user roles, and the granularity and duration of control. An object-specific "fair dragging" policy is proposed, where a user can obtain mutual exclusive control over a resource object only while holding the mouse button over the object. Fair dragging qualifies as an interactive, uniform, fine-grained, and short-term policy. Crowley et.al. [43] distinguish between four policies: no floor; implicit request and grant; explicit request and implicit grant; and explicit request; explicit grant. All are user-interface centered.

We propose a refined model of policies, taking into account queuing and preemption or requests. Policies are either explicit, requiring a distinct action or signal from a user to impact the control mechanism, or implicit through observation of system events and conditions. We also distinguish between policies enacted by users through the graphical user interface (GUI), versus host-enacted and network-centric policies. In *chaired* control, a specific host is elected as permanent moderator over specific resources in a session, acting as arbiter over floor assignment. *Roving* control passes the privilege to arbitrate among session members, e.g., by equating the floor coordinator FC with the floor holder FH and requiring users to address floor requests to the current FH. *Task-oriented* control assigns a floor to execute one or more steps to conclude a task and revokes the floor automatically at

completion time. *Activity-oriented* control uses specific keys, voice activation [77], movement of objects, gestures, or other input to trigger a floor handover. *Timed* control defines a duration or timeout on a floor, measured by a logical clock, system clock, or session events.

Scheduled control passes floors among participants according to some agenda, such as round-robin, varying a token passing order defined by an underlying mechanism. *QoS-based* control takes into account the Quality-of-Service demanded by a user, giving for instance preference to service requests that require less bandwidth. *Voting*-based control [86] uses a consensus procedure among hosts to determine the next floor holder. *Election*-based control [85] lets users draw a ticket defining a service order from a master site and receive floors in according sequence, which can be varied with a lottery-based allocation [228]. In *queued* control, multiple floor requests are placed in a waiting buffer and serviced according to a queuing policy, such as Least-Recently-Served, Least-Frequently-Served, First-Come-First-Served, or Shortest-Task-First, if the time to execute tasks can be anticipated. Timestamps or sequence numbers attached to floors can be used as criterion to determine a service schedule. Since sequence numbers are not unbounded, numbers older than a threshold are recycled to avoid unintended wrap-around. Finally, in floor *brokerage*, each user may bid on resources allocated by a resource broker and spend currency for the usage of a resource, where less aggressive resource consumption may gather merit credits.

In the absence of a service policy, a "free for all" or *no floor* scheme permits resource access on a first-come-first-served basis with the risk of conflict. Specific policies may be appropriate for certain resources, but ill-suited for others, and hence be adapted to a given interaction modality. For example, a predefined ordering scheme is suited for instrument control, but not for unpredictable turn-taking in conversations. The size of queues must be limited relative to the session size, since it does not make sense to allow a long backlog of requests and hence large delays in interactive work. Policies must be fair, i.e., allocate floors to session members evenly and consistently, in relation to the session purpose and user roles. Different policies are used to manage different resources. The choice of a combination of mechanism and policy depends on the resource. For instance, an audio channel with few

users is self-moderating and has no notion of consistency, whereas larger sessions and a combination of audio with documents enforces synchronization constraints. In the next section, we present a solution to integrate floor control in the process of user-specified Quality-of-Service (QoS) tuning.

3.2.3 Quality-of-Service Mapping

In a large number of collaborative multimedia tools developed for the Internet, users cannot specify QoS requirements in access and presentation of resources. Alfano and Sigle developed a model [10] to integrate user-level and system level QoS-control in collaborative environments. Amir *et al.* [12] discuss the SCUBA protocol, which implements a feedback scheme on media consumption based on discovery and adaptation to receiver interest. Floor control input can help the mechanism decrease the convergence time to ready the next bandwidth partition. SCUBA augments floor control because media streams can be throttled in accordance with who holds the floor in a session, achieving better network utilization. We discuss how floor control can be intertwined in the QoS negotiation or renegotiation process, since QoS requirements for users are likely to shift throughout a session depending on their work focus. Specification of QoS requirements describes user interest for a specific media quality in isolation, and in relation to the other media in use. Furthermore, individual QoS choices may impact the reception and presentation QoS of other users, requiring a mechanism to establish collective choice for QoS parameters and for combined media quality characteristics. An abstraction of layered QoS mapping of user requests to resources is shown in Figure 3.3.

In order to merge QoS tuning under multiple constraints with floor control, we characterize the QoS for each resource as a vector

$$Q_\rho = (\mu_1, \mu_2, \ldots, \mu_l) \tag{3.1}$$

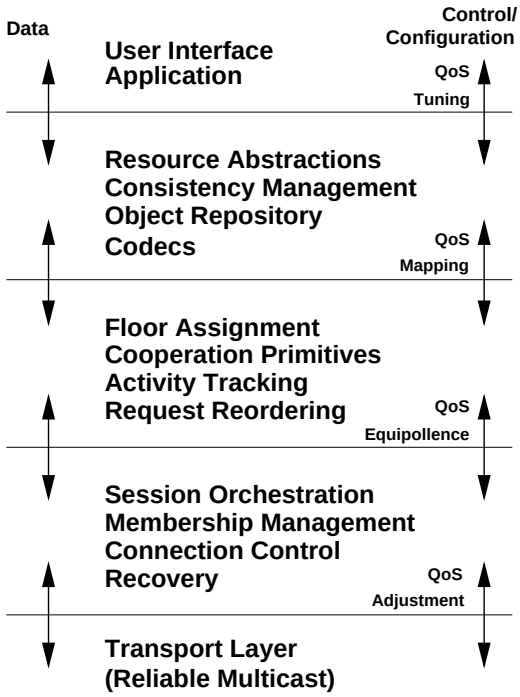

Figure 3.3: QoS mapping with floor control input.

where $\rho = \{Video, Audio, Image, Graphics\}$ denotes the media type and $\mu_i(\rho), i = 1, \ldots, l$, represents a list of media-specific QoS attributes. Table 3.1 depicts a list of possible attributes for generic media types.

ρ	$\mu_i(\rho)$
Audio	Samplerate, samplesize, lossrate, delay, fidelity
Video	Framerate, framesize, codec, color, resolution, jitter, delay
Image	Size, resolution, color, codec, compression
Graphics	Size, dimensionality, resolution, colors, format

Table 3.1: Select QoS characteristics for various media types.

We define the *characteristic QoS* (cQ) of a resource R as

$$cQ_R = \sum_{i=1}^{l} \omega_i \mu_i \tag{3.2}$$

where ω_i represents a weight factor for each media attribute determining the quality level for each attribute, such that $\sum_{i=1}^{l} \omega_i = 1$. Depending upon the application type and performed

tasks, users' expectations may focus on high throughput (e.g., a high video frame rate) or low delay (e.g., for audio), and accept tradeoffs such as a certain degree of lossiness or jitter. This allows for a modular, normed representation of different QoS for all types of resources and fast evaluation of QoS conditions. The actual characteristic Q^a in an active multimedia session is monitored by a background process correlating the current values μ_i^a with current host and network conditions such as CPU load and throughput. Q^a is then compared with the QoS Q^d required by the session specification, or demanded by the user, resulting in the *feasible* quality Q^f, where

$$Q^f = Q^a - Q^d \tag{3.3}$$

If a parameters $\mu^f > 0$, the resource is underspecified and can be scaled up [213], i.e., the user asks for less than the QoS mapping function is able to guarantee, and a floor request on this resource can be satisfied with the demanded media quality. Otherwise the floor request cannot be met and must be scaled down, with the option to discard the request or present a subset of attributes in Q^f to the user asking for this floor for this resource, to adjust the bias for the media attributes accordingly. The feedback process between system and user requires that the parameters for attributes are percolated up to the user interface and presented in an intuitive way. This process of cooperative media scaling based on QoS specifications represents a tangible interface for users to fine-tune their cooperation in relation to each other, as well as host and network conditions.

3.2.4 User Perspective

Few end-user studies have been conducted on live interaction, constrained by the available hardware and collaboration technology. Consequently, little information exists on how to tailor coordination software and collaborative interfaces to end-user needs. User interaction relies on and is strongly affected by the initiation and enactment of group coordination, as well as the dissemination and presentation of control information. To date, no guidelines exist on how to develop collaborative user interfaces. Group coordination is affected by technical aspects such as heterogeneous hardware or multimodal interfaces, as well as

sociological aspects of interaction such as culture and group dynamics [96]. Hence human factors plays an important role in the design of group coordination systems. It is primarily the addition of various user dimensions in the interaction process that sets floor control apart from concurrency control.

Interface

User interface design is more complex, because local and remote information, private and shared, must be condensed and represented in a way that does not overwhelm users, and manifests awareness and presence in live interaction. "Every access to every (shared) object should be checked for current authority" is the axiom of *total mediation* [202], however, it is unclear how tangible or transparent floor control mechanisms can be made available to users without creating a notion of intrusion, obstruction, or serialization constraints on joint tasks that may lead to rejection of such a mechanism. Users should have the option to activate or deactivate sharing on resources at various levels of granularity, based on roles, capabilities, and authorizations, collaborate anonymously, receive updates on activities of collaboration partners and resources, and be able to poll the floor control status for local and remote resources. Furthermore, floor control should be configurable at run-time, and allow for collaborative undo [176] or redo based on the collaborative history for reversible or replayable activities.

User interfaces should integrate local and remote views, in public and private windows on the local desktop and provide a notion of panoramic view on the shared workspace. Direct manipulation "WIMP" (windows, icons, menus, pointers) interfaces are not ideal for this purpose, because they provide a flat metaphor for three-dimensional workspaces and clutter the desktop with icons and windows. They are also ill-suited for collaboration across wearable platforms, where users cannot handle high cognitive load in the interface in order to manipulate data. Multi-scale interface [80, 184] which allow users to navigate and zoom within a three-dimensional rendition of the shared workspace, may evoke better focus and a spatial awareness to shared activities. Supplying a feedback mechanism on

current floor activity in relation to resource activity permits use of the *reciprocity* rule [201], which, translated into the CSCW context, states that local-to-remote activities should result in a remote-to-local feedback. Local and remote workspaces, in their entirety or with regard to specific resources, should be independently viewable by users, or displayed in synchrony if needed. Ultimately, users judge a protocol by its relevance with regard to improved productiveness and ease of use, which should headline design efforts for floor control protocols.

When a user initiates or joins a session, floors for each available shared resource are created. A control mechanism must also accommodate the ad hoc introduction of new resources in the workspace. State information for the shared resources is disseminated to other session nodes or a floor server. In the distributed case, hosts may achieve consistency by exchanging states directly with all members in the multicast group comprising a specific resource, or through a diffusing computation, where each node communicates only with its direct neighbors. During session conduction, coordination primitives such as floor requests are multicast to the respective groups, querying and updating their local state tables. A user should be able to obtain information on the local floor state table and updates through the GUI. The floor control protocol must ensure that floor requests and transfers do not cause deadlocks, starvation of a user, or unfairness.

When a resource is removed from the shared space, the adjunct floor is removed. In case that a user withdraws or a host fails, *orphaned floors* for resources located in the network or a different end-host must be reassigned. A floor control protocol must also adapt to the eventual loss of a FC or FH and swiftly shift roles to other users using, for example, an election mechanism. A split session may continue, if orchestration and coordination within the session partitions are consistent and users choose to not abort the session. Session mergers must also be accommodated by the control protocol joining control tables and recreating consistent state within joined multicast groups.

Deployment

Floor control should be transparent in function, but tangible in status, using multimodality in input and output on control states. For instance, floor requests can be activated with a data-glove gesture, voice, or by simply pressing a mouse button, and feedback on floor states can use color, sound, icons, or haptic feedback. Cognitive cues such as transparency or color in the visual representation of a shared resource are helpful to indicate its current floor state: an opaque or green rendition of a widget could depict a locally held floor, a light-shaded or yellow rendition indicates that a floor is requested by a remote site, and a transparent or red object icon signifies that floor is held by a remote site. Auditory cues can also be used as support, and pointing on a locked resource could hence trigger an auditory signal to depict the sharing status.

A common solution to mark multiple floors for the same visual resource, e.g., telepointers, is to tag them with name labels or colors identifying their holders, although this approach is only practical for small sessions due to limited screen real-estate. The GUI must mark or block a resource, whose floor is in transition to a new FH, to avoid inconsistencies or multiple assignment of the same floor, e.g., by framing a resource object yellow or red. Floor policies should not depend on specific user interface modalities or properties of a specific host platform, because GUI design and hardware specifics are subject to constant evolution. Floor states can be coupled with window states, e.g., the iconization of a window could disable all floors for resources accessible through that window.

The main advantage in making floor control a network service lies in rapid development of collaborative systems, by providing an application-programmer's interface or library of floor management routines, and gaining a universal control infrastructure, across which applications can coordinate each other. Table 3.2 summarizes the calls that applications may use to interact with a floor control protocol, handling a specific floor F for a resource R of user U in a session S.

Call	Semantics
`AddFloor`	add template for F to active floors
`AllFloor`	display all active floors for R, U, or in S
`AssignFloor`	assign F to another U without request
`CancelFloor`	stop request for F
`CreateFloor`	bind new F uniquely to R and U
`ExpandFloor`	enhance scope of F to multiple objects
`FreezeFloor`	freeze usage of F on R
`GrantFloor`	grant F for R to U
`TrackFloor`	display collaborative history for F
`InfoFloor`	query current attributes and usage state for F
`KillFloor`	eradicate active floor(s) for specific R or U
`PauseFloor`	set active F idle
`ReleaseFloor`	free F after turn completion
`RelinquishFloor`	give up F after revokement
`RemoveFloor`	remove inactive F
`RequestFloor`	request idle or used F and enter queue
`ResumeFloor`	reactivate sleeping F after idle period
`ResetFloor`	reset floor attributes for F, U, R, or entire session
`RevokeFloor`	force release of F from U
`ShrinkFloor`	reduce scope of F to fewer users

Table 3.2: Floor control API primitives and functional semantics.

3.3 Efficacy of Floor Control Protocols

We propose a taxonomy of floor control protocols based on their operational principles, and compare their performance ("efficacy") taking into account user behavior and generic system properties. The performance is correlated with the responsiveness and scalability of various floor control mechanisms by taking into account the processing overhead encountered per turn. Based on our results, we make the case for hierarchical floor control, operating on a control tree, whose logical topology resembles the backbone reliable multicast tree and the actual underlying multicast routing tree. Our objective is to provide a new methodology in the evaluation of coordination protocols, and to explore new avenues in floor control scalability through conceptual integration with underlying multicast transport strategies.

3.3.1 Taxonomy of Floor Control

Three properties forming the operational pillars for a floor control protocol are: (1) the random or scheduled access to resources, characterized by the mechanism and node topology,

by which floor information is probed or relegated via directives between nodes; (2) the centralized or distributed establishment of control among the nodes in the session; and (3) the policy, by which floors are commonly processed among the nodes. Property (1) is the major design decision for a group coordination protocol, and determines, which control and logical roles as well as policies are established in a session. This property lays the foundation for the taxonomy shown in Figure 3.4. Known paradigms of floor control protocols are divided here into two classes: *Random-access Floor Control* (RFC) and *Scheduled Floor Control* (SFC) protocols. RFC protocols lets users contend for the shared resource by sensing its status remotely before or during resource access. In RFC protocols, users contend for the opportunity to attain the floor for a shared resource. SFC protocols use token passing, reservations or polling as access mechanism in a session. A dashed line shows a protocol whose performance will not be discussed in this paper. This taxonomy is not exhaustive, but serves as a first attempt to characterize predominant solutions on floor control.

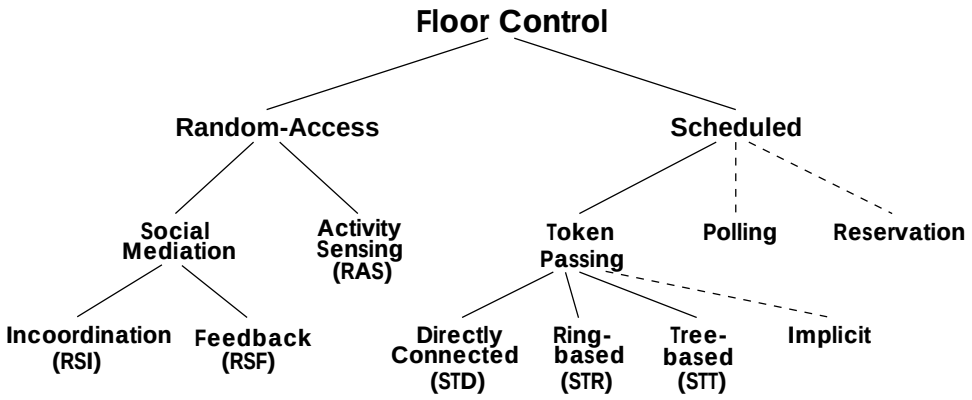

Figure 3.4: Taxonomy of floor control.

RFC protocols are based on the concept of sensing the status of a remote resource, which can be accomplished either by users tracking each other's activities through the user interface, or by the system, sensing the state of an application, host, or network for local and remote activities. The global floor state is marked with assertions on local variables and no token entity is explicitly exchanged as placeholder. RFC schemes are inherently

contention-based, because nodes must actively compete for a floor. The collective state of local assertions must be kept globally consistent. Continuous sensing of many remote resources' states, either by man or machine, can be costly, and there is a higher likelihood of collisions due to network latency or lack of coordination.

In contrast, SFC protocols rely on the exchange of a unique and explicit floor token, with nodes either being polled by a coordinator node for pending control messages, or with the token being passed in the order prescribed by the logical geometry of the session. The uniqueness of the token guarantees exclusive usage of the floor. Resource contention is hence dispersed by regulated floor capture. Control directives are used to request, deny, reserve, or grant a floor. Using explicit floor tokens does not mean that users have to make a conscious effort to signal reservation and exchange of such tokens. Rather, the floor control subsystem orchestrates these electronic placeholders as a form of concurrency control, although users can influence or modify system decisions. SFC comes in two flavors: nodes can proactively send control messages to "ask" for the floor, or they can wait passively, until the floor is being "offered" by polling or passing through. A shortcoming of SFC schemes is the cost incurred in tracking floor tokens and ensuring their uniqueness and authenticity. SFC schemes generally operate on a specific infrastructure such as a ring or tree, which logically organizes the session.

Figure 3.5 shows a prototypical call sequence for two users U_1 and U_2 communicating with a floor coordinator FC, assuming that U_1 has precedence over U_2. In RFC, using implicit floor control, `REQ` (request) equals "sense resource state", `GRT` (grant) equals "active on resource", `WAIT` equals "pending", `ACK` (acknowledge) is equal to confirmation of user activity through an external indicator such as a busy-floor icon, `DNY` (deny) correlates with "busy", and `REL` (voluntary release) and `RLQ` (forced relinquishment) is equal to "free resource". First, U_1 requests the floor from FC, which grants it after checking whether the floor is free. U_2 sends a request shortly thereafter, and is told to wait, until U_1 confirms floor reception. After such confirmation, U_2 is denied the floor, and after U_1 had its turn and releases the floor, FC assigns the floor based on the queued request from U_2, which

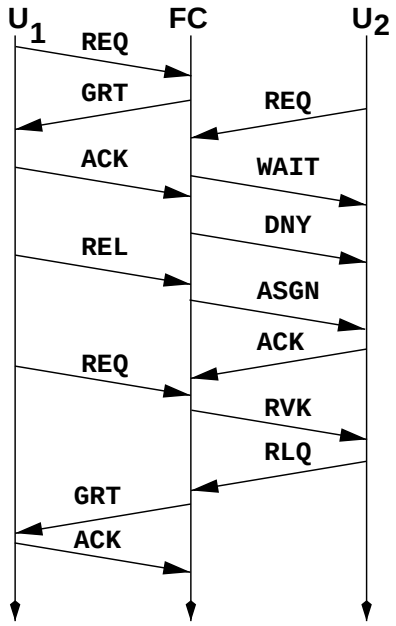

Figure 3.5: Call structure in SFC.

sends an `ACK` back. U_1 requests the floor again, and FC revokes the floor from U_2, who relinquishes it. FC grants the floor to U_1, who acks reception. A chaired, queued, preemptive policy is enacted in this dialogue. If requests arrive at FC at the same time, the timestamp or sequence number of the arriving control headers can be used as a criterion to sequence requests.

Figure 3.6 shows alternative coordination geometries directing the flow of control information between nodes. In case (a), four users attempt access to a central shared resource R (circle) either managed by some distinct FC node, or with direct uncoordinated access to R. Case (b) depicts floor control on replicated resources in a fully-connected network, where users communicate directly with each other and one users assumes the role of the roving FC. Case (c) shows a ring topology, where floors are being passed round-robin from user 1 to 4. Floors can only be acquired, when the token passes by. Case (d) depicts a star-geometry with central coordination, where resources are replicated and a users can only acquire access to remote resources by asking FC. Case (e) conceptualizes a tree-based floor control mechanism, in which control primitives are forwarded between levels toward the current FC, which can be any node in the tree. Topologies can be derived from the ac-

tual underlying network connectivity, or are logical artifacts to structure the flow of control messages between nodes. Large-scale collaboration architectures may use a conglomerate of various logical topologies, for instance a bus in the local network, a ring for the metropolitan scope, and a tree for long-distance interconnection.

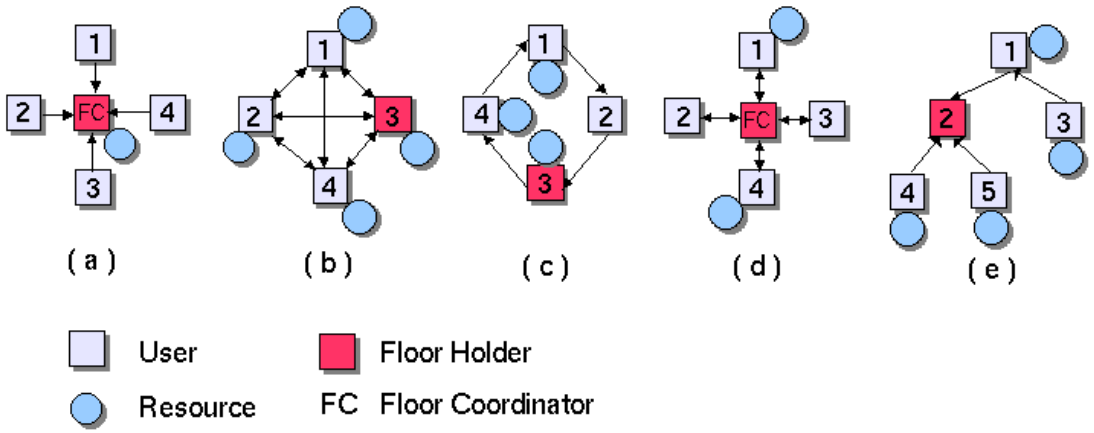

Figure 3.6: Coordination topologies.

3.3.2 Comparative Analysis

In this section we compare the efficacy of floor control protocols to disseminate control information and warrant fluid floor passing. Little related research has been published on this subject. Pendergast [174] argues, based on message and latency comparisons for generic collaboration environments, that an independent-objects model achieves the best performance, followed by central sequencing and distributed operation. Distributed algorithms should only be used when response time and physical or temporal ordering are required. The study assumes generic state machines for modeling the three application types and does not take network or user behavior into account. Ahuja *et al.* favor single-site approaches in comparison with multi-site and hybrid approaches [7]. Trossen and Katona [223] show the limitations in scalability of T.120 with a conference load model evaluating floor passing and asking [52].

Our analytic comparison of known classes of floor control protocols is a first attempt to characterize the efficacy of interactive behavior of people and processes from a resource contention perspective and to highlight performance differences among coordination schemes. The intention is not to predict exactly how a protocol performs for a given *CE*; accomplishing this would require far more host- and network-specific details and statistics on user behavior. Our goal is simply to assess the overhead of various protocols with regard to control state management. The basic methodology is derived from multiaccess communication, based on the analogy between access mitigation for the data channel and shared resources.

To make the analysis tractable, we state the following assumptions: the individual host processing cost for control packets, including protocol overhead and user-interface specifics, is the same for all hosts; message delivery between hosts is reliable and no failures in hosts or the network occur; we only account for the processing effort in sending floor control messages; information from a station reaches any other station with the same average system-wide propagation delay; the interarrival rate of floor requests is Poisson, given that there is no indication for cross-correlations between subsequent floor requests; depending on the session model, subsequent requests are either being discarded, or queued and served in FIFO order; finally, the task length, i.e., the floor holding time, is normalized.

For the analysis, we take into consideration the interarrival rate for floor requests, the task length, and the network propagation delay. We consider a point-to-point model of message dissemination, as well as a broadcast model, because a number of systems have been implemented for one model, but not for the other. The broadcast model uses IP-multicast, i.e., a hosts needs to send a message only once to the network interface, where it is multicast to all the receivers.

The notation used in our analysis is summarized in Table 3.3: to model acquisition of a floor token ahead of time, we introduce the idea of "think time", denoted with β_1 (the time until the expected floor arrives at the local station) and β_2 (the time once the floor token is present at the local station and offered to the user); γ is the average time to process or communicate a floor directive; δ is the duration of an activity period, which is normalized

β_1	Average "think" time before floor token arrival
β_2	Average "think" time at floor token presence
γ	Average processing time for a floor directive
δ	Duration of average activity period
$n\epsilon$	Processing and unicasting overhead to n receivers
η	Efficacy of a floor control protocol
G	Average offered floor request load
ι	Average duration of idle time
λ	Floor request interarrival rate
m	Average number of stations in session
n	Average number of active stations in session
ν	Average vulnerability period
τ	Average propagation delay

Table 3.3: Analysis parameters.

to 1; we use $n\epsilon > \tau$ as the additional delay to account for the dependency of the communication and processing overhead on the number of stations collaborating; $G = \delta \times \lambda$ is the normalized offered request load on floors, including new and previously denied and resubmitted floor requests; ι sums up the expected idle time for a resource during floor holding time; λ represents the relative frequency of demand for resource access and thus indicates the contention level; $m < \infty$ stations denotes all stations being a member in a session; $n < m$ is the number of active stations in the multicast group for the current floor and transmission; ν is the time during which a station's attempt to access a resource can be intercepted by another station; and τ denotes the average end-to-end delay between stations, including packetization delay, generic queuing transmission times, and local processing overhead. The activity period and time to process and communicate floor directives are assumed to include the time incurred in providing feedback among stations.

User behavior in terms of the switching of control over a particular resource is incorporated into this analysis with a turn taking model [55], which serves as the conceptual blueprint for our analysis. A prototypical turn taking period is depicted in Figure 3.7, which shows the call pattern between stations and the effect on the particular controlled resource. A turn has accordingly three stages: the contention time X, accounting for request, contention and granting or denial periods; an activity time A, accounting for floor holding and releasing; and an idle time I (which may or may not occur) accounting for the time period when the resource is not actively being used.

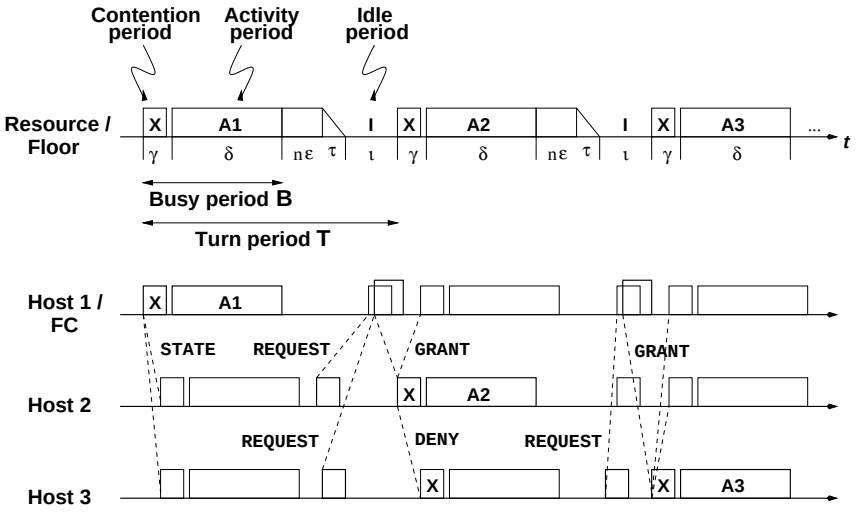

Figure 3.7: Turn taking periods for a resource and three stations.

Figure 3.7 depicts a conversation between three stations trying to access a resource. First, station 1, which is also FC, holds the floor and its activity outcome is transmitted to the other stations. Second, stations 2 and 3 try to acquire the floor, and station 2 wins, because its request was first received at FC. Station 2 starts accessing the resource, until the floor holding time expires. Third, station 3 acquires the floor without collisions. This diagram represents the case when FC is separate from FH, and is applicable to both implicit and explicit floor passing. It is the variations in allocation patterns and duration of floor periods that make certain control mechanisms more efficient than others. Based on this abstraction, we define the *efficacy* of a floor control protocol, denoted by η, as the proportion of time that a protocol needs to allocate a resource, including overhead from the protocol itself, the network, and user behavior. In other words, we want to assess the reactiveness of a protocol in a specific system architecture to signal for attention, submit a request, receive a reply, and select a user to be in charge of a resource. Formally, the efficacy η is the ratio of the average floor usage time $\bar{U}$ vs. the overall average turn length $\bar{T} = \bar{B} + \bar{I}$, with an average busy period $\bar{B}$ and idle period $\bar{I}$:

$$\eta = \frac{\bar{U}}{\bar{T}} = \frac{\bar{U}}{\bar{B} + \bar{I}} \tag{3.4}$$

The average contention period $\bar{X}$ and activity period $\bar{A}$ together form the average busy period, $\bar{B} = \bar{X} + \bar{A}$. These turn periods serve as the building blocks for our efficacy analysis. An important aspect of our comparative analysis is the impact that multicasting at the network layer has on the efficacy of the floor control protocols. When network multicasting is not available, the user stations are forced to contact one another explicitly, which substantially increases the processing overhead at stations and the possibility of disagreement on which station has the floor.

We model the existence of multicasting at the network layer by assuming that a user station needs only to pass information once to the network (during an activity period and to gain the floor) for the information to reach all other stations with the same average delay. In the following paragraphs, we discuss the efficacy of prevalent solutions with and without multicast support.

Random-Access Group Coordination Schemes

Incoordinated Social Mediation (RSI) refers to a purely random scheme of resource access in a self-moderating session. Shared access becomes guesswork and data inconsistencies are expected. Possible causes are insufficient feedback through the user interface about remote activities, delay caused by host or network limitations, or uncooperative users on a self-moderating channel. As a consequence, contending stations, being unaware of each others' immediate actions, may experience floor acquisition conflicts by trying to access a resource at the same time. Although all information is distributed reliably to all user stations, the latency introduced by the system can lead to inconsistent views of the floors at different stations. When this occurs, we assume that all users involved in the conflict must restart their activities. There is no system support to mediate conflicts and ensure system-wide consistency of the state of the shared resource, and all messages must hence percolate through the entire protocol stack to be reflected in the user interface, which makes the conflict detection process long and inefficient.

Theorem 1 *The efficacy of RSI without multicast support is*

$$\eta_{RSI} = (\delta + n\epsilon)\lambda e^{-2\lambda(\delta+n\epsilon)} \tag{3.5}$$

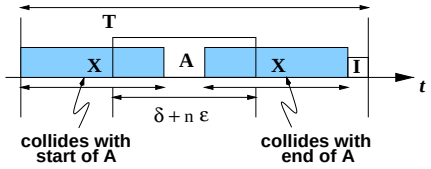

Figure 3.8: Prototypical RSI timeline.

Proof: Figure 3.8 shows a prototypical RSI turn. The proof is the same as for medium access in an ALOHA channel [25]. In the point-to-point model, we assume that n stations are actively monitoring each other, and it takes an additional time $n\epsilon$ for all stations to perceive the activity from a given station. All the information is exchanged reliably, and the average vulnerability interval is twice the total of the task length and the added overhead incurred in serial communication of the task information to all stations (i.e., $\delta + n\epsilon$), because messages can intercept activities any time. Because request arrivals are Poisson, the probability that a task is successful is $e^{-2(\delta+n\epsilon)\lambda}$. The success probability times the number of arrivals in one activity period results in Eq. 3.5. □

Corollary 1 *The efficacy of RSI with multicast support is*

$$\eta_{RSI}^{MC} = \lambda\delta e^{-2\lambda\delta} \tag{3.6}$$

This follows under the assumption that every update to and from a station requires only one transmission.

Social Mediation with Feedback (RSF) assumes cooperative users following agreed-upon social protocols, relying on feedback support on remote activities from the network and user interface. If a user contends for a floor and perceives remote activity, she would back off for a random short period and attempt to reclaim the floor after remote activity subsided. RSF in video conferencing, as with the MBone tool vic [150], is often realized as "voluntary distributed control" [77], where cooperative users switch video streams manually on and off, depending on whether they prefer to receive or send specific video transmissions. Analog

POTS conferencing also relies on RSF, which works well for very small groups. Except for user interface updates, both RSF and RSI incur little implementation cost regarding user coordination.

Theorem 2 *The efficacy of RSF without multicast support is*

$$\eta_{RSF} = \frac{\delta}{\delta + e^{2\lambda(\gamma'+n\epsilon)}\left[\frac{e^{\lambda(\gamma'+n\epsilon)}-1-\lambda(\gamma'+n\epsilon)}{\lambda(\gamma'+n\epsilon)(1-e^{-\lambda(\gamma'+n\epsilon)})} + \gamma' + n\epsilon + \tau + \iota + \frac{1}{\lambda}\right]} \tag{3.7}$$

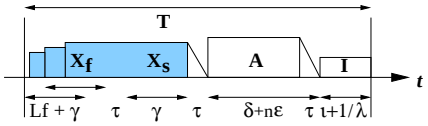

Figure 3.9: Prototypical RSF timeline.

Proof: A prototypical timeline of RSF is depicted in Figure 3.9. n active stations need to sense and process remote activity through the network interface. The success probability, denoted as P_s, equals the probability that no activity packet arrives in an average vulnerability period ν of $2(\gamma' + n\epsilon)$ s, i.e., $P_s = P[0\ packets\ in\ \nu] = e^{-2\lambda(\gamma'+n\epsilon)}$. The average utilization period lasts $\bar{U} = \delta P_s$, and the length of the average busy period $\bar{B} = \bar{X} + \bar{A}$ is determined by the time needed to handle unsuccessful floor requests in the failed contention period X_f and successful requests in X_s, with $\bar{B} = (1-P_s)X_f + P_s X_s$. An average failed turn attempt consists of a geometrically-distributed indefinite number ($\overline{L}$) of interarrival times of floor requests with duration $\overline{f}$ s(econds) (average time between failed floor-request arrivals), plus the duration observing a request (γ'). The values for $\overline{L}$ and $\overline{f}$ have been derived by Takagi and Kleinrock [217]. Substituting our notation in these results, we obtain $\overline{L} = e^{\lambda(\gamma'+n\epsilon)}$ and $\overline{f} = (\lambda(\gamma' + n\epsilon))^{-1} - e^{-\lambda(\gamma'+n\epsilon)}/(1 - e^{-\lambda(\gamma'+n\epsilon)})$, respectively. Accordingly, the average time of a failed turn attempt equals $X_f = \left[\frac{e^{\lambda(\gamma'+n\epsilon)}-1-\lambda(\gamma'+n\epsilon)}{\lambda(\gamma'+n\epsilon)(1-e^{-\lambda(\gamma'+n\epsilon)})}\right] + \gamma' + n\epsilon + \tau$. An average successful turn lasts $X_s = \delta + \gamma' + n\epsilon + \tau$. Finally, based on the Poisson assumption, an idle period consists of an average idle time interval plus the time until the next floor request arrives on the average, $\bar{I} = \iota + \frac{1}{\lambda}$. Substituting into Eq. 3.4, we obtain Eq. 3.7. □

Corollary 2 *The efficacy of RSF with multicast support is*

$$\eta_{RSF}^{MC} = \frac{\delta}{\delta + e^{2\lambda\gamma'}\left[\frac{e^{\lambda\gamma'}-1-\lambda\gamma'}{\lambda\gamma'(1-e^{-\lambda\gamma'})} + \gamma' + \tau + \iota + \frac{1}{\lambda}\right]} \tag{3.8}$$

With multicast support, the vulnerability period reduces to $2\gamma'$, and ϵ becomes negligible.

In *Activity Sensing* (RAS) [41, 84], activities on shared resources are monitored by a background process at the session layer (without the user having to do so) in order to sense, which node currently operates on the resource. The RAS concept is related to collision sensing on a multiaccess channel [25]. In principle, no changes to a collaboration-unaware application are required to integrate floor control by modifying the X-server to intercept and filter calls to the application signifying access to a shared resource. Similar to the socially protocolled RSF, a RAS system agent would back off when detecting remote activity, deny the local user the floor until remote activity subsides, and signal a free floor afterwards. By that, a distributed collective of activity sensing agents enacts more accurate monitoring about resource states than humans could deliver. The disadvantage of this scheme is its high implementation cost and its reliance on short link latencies, as shown below, which makes it only suitable for LANs.

Theorem 3 *The efficacy of RAS without multicast support is*

$$\eta_{RAS} = \frac{\delta}{\delta + e^{2\lambda(\gamma'+n\epsilon)}\left[\frac{e^{\lambda(\gamma'+n\epsilon)}-1-\lambda(\gamma'+n\epsilon)}{\lambda(\gamma'+n\epsilon)(1-e^{-\lambda(\gamma'+n\epsilon)})} + \gamma' + n\epsilon + \tau + \iota + \frac{1}{\lambda}\right]} \tag{3.9}$$

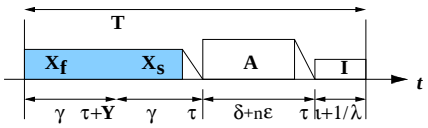

Figure 3.10: Prototypical RAS timeline.

Proof: The timeline is shown in Figure 3.10. Without multicast support, a station has to send floor directives individually to every other station, which according to our assumption

takes $\gamma' + n\epsilon$. Because multiple unicast messages are used by each station, floor-state inconsistencies can arise during the entire time. The station is exchanging floor directives, i.e., the vulnerability period of the protocol is $\nu = 2(\gamma' + n\epsilon)$. Using this vulnerability period, Eq. 3.10 follows using the same approach as described in the proof of Theorem 3.7.

Corollary 3 *The efficacy of RAS with multicast support is*

$$\eta_{RAS}^{MC} = \frac{\delta}{\delta + \tau + \frac{1}{\lambda} + e^{\lambda\tau}(\gamma' + 2\tau + \iota)} \tag{3.10}$$

Proof: The access strategy is assumed as nonpersistent, i.e., a station backs off immediately from attempting to access a resource and claims the resource once it appears free again. The vulnerability period for accessing an unused resource is one propagation delay, $\nu = \tau$, within which other stations can cause a conflict (opposite to RSF, where twice the length of the contention interval is the average vulnerability period); therefore, $P_s = P[0 \textit{ packets in } \nu] = e^{-\lambda\tau}$. The average utilization period is $\bar{U} = P_s\delta$. The average length of a successful busy period is simply $\gamma' + \delta + 2\tau$, which accounts for the delivery and processing of floor directives, the activity period, and associated network latencies. The length of an average unsuccessful activity period consists of one truncated activity lasting γ' s, followed by one or more similarly truncated activities sent within time Y s, where $0 \leq Y \leq \tau$. The expected value of Y is [217] $\bar{Y} = \tau - \frac{1}{\lambda}(1 - e^{-\lambda\tau})$; therefore, the average duration of a failed contention period is $\gamma' + 2\tau - \frac{1}{\lambda}(1 - e^{\lambda\tau})$. The length of the average busy period is then $\bar{B} = \gamma' + 2\tau - \frac{1}{\lambda} + e^{-\lambda\tau}(\delta + \tau + \frac{1}{\lambda})$. The average idle interval is again $\bar{I} = \iota + \frac{1}{\lambda}$. Substitution into Eq. 3.4 yields Eq. 3.10. □

Scheduled Group Coordination Schemes

In SFC protocols, floor token transmission must be reliable to ensure that floors are not duplicated, lost or forged. We exclude such situations from our analysis. In contrast to the contention time γ' in random-access schemes, γ denotes here the time to transmit a floor token to the next station. We assume for all protocols that τ signifies the average

propagation delay for multiple routing hops between stations coalesced into one hop. A packet must hence traverse on the average the same number of hosts on the path from the sender to a group of receivers.

We can identify three major SFC solutions to floor control: In *token passing* the floor is being asked for or offered to the stations in the session in a predefined service order. Stations, active or inactive, are either directly connected to each other, or they are logically arranged in a multihop ring or tree geometry. *Implicit token passing* [25] is a special case used to reduce token size, in which a station relinquishing the floor simply goes idle. The next node in sequence detecting resource idleness takes the floor if a local request has been recorded, or remains quiet. Similarly, the other nodes wait for increasingly longer timeout periods to detect idleness and take the floor if requested. However, such collision avoidance lacks fairness in accessing tokens, because stations further away from the current FH may never attain the floor.

In *Direct Coordination* (STD), each station in a group is fully connected to every other station. STD improves the response time. However, the number of links is $\frac{n(n-1)}{2}$ and grows as the square of the number of nodes in the session, which makes this solution unscalable, in particular for unicasting. Messages must be ordered or a voting mechanism among nodes must determine a FH successor. Furthermore, cognitive abilities of human users to handle flows from all participants are limited. Many small-scale commercial video conferencing systems follow this model.

Theorem 4 *The efficacy of STD without multicast support is*

$$\eta_{STD} = \frac{\delta}{n(\gamma + \tau + \epsilon) + \delta + \iota + \frac{1}{\lambda}} \tag{3.11}$$

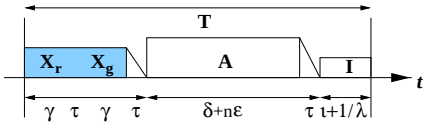

Figure 3.11: Prototypical STD timeline.

Proof: The timeline for STD is shown in Figure 3.11. The average utilization period is $\bar{U} = n\delta P_s$, because floor capture is perfect and any one of the n active stations can acquire a floor of holding time δ with success probability $P_s = \frac{1}{n}$. A floor may not be released at FH, until successor FH' received it. The control packet overhead is γ and the propagation delay is τ. Unicasting a floor request to the $n-1$ active nodes amounts to $(n-1)(\gamma+\tau+\epsilon)$, plus $(\gamma + \tau + \epsilon)$ for a reply. The average activity duration is $\bar{A} = \delta$ and may be trailed by an idle interval consisting of a period ι and an average interarrival time for all nodes, $\bar{I} = \iota + \frac{1}{\lambda}$. Substitution into Eq. 3.4 results in Eq. 3.11. □

Corollary 4 *The efficacy of STD with multicast support is*

$$\eta_{STD}^{MC} = \frac{\delta}{\delta + 3(\gamma+\tau) + (n-1)\epsilon + \iota + \frac{1}{\lambda}} \tag{3.12}$$

With multicast support, the request-reply-release exchange of control packets takes a time of $3(\gamma+\tau)+(n-1)\epsilon$, if we assume that every station incurs host processing overhead from its $n-1$ neighbors.

In *Ring-based Coordination* (STR), stations in a session form a logical ring, and a floor token cycles through the ring in a round robin order. STR is particularly suited to ensure totally ordered and atomic delivery. Pendergast [174] discusses a group support system operating on a token ring. Ziegler *et al.* [240] analyze packet-switched voice conferencing mechanisms in a logical ring. A station that is ready to start an activity captures the passing token, inserts a command sequence with address and control information, sends the activity packets within this turn period, and transfers the token after completion to the successor station. A station without pending floor requests passes on the offered token. If a token is held for excessive time it can expire, or a busy-token for this floor is sent across the ring to indicate that the floor is taken and the token "alive". A station waiting for the floor can insert a reservation tag into a busy token to mark that it is next in line for holdership. The circulating token hence replaces request and grant messages as a form of

explicit signaling, and contention among nodes in the ring is based on claiming the roving token.

A related case is floor control over a token bus, where a node passes the floor token along the list of stations attached to the bus. The delay in a token bus is inherently larger compared to a token ring [25] and will not be discussed. A shortcoming of STR is that a predefined token passing schedule does not reflect spontaneous interactivity. There are many variations on the detailed operation of floor control in a ring structure, concerning acquisition and release of the floor token. The floor can be granted ahead of its position to a successor station, or it may only be acquired by a station when it passes through that station. Likewise, a token can be immediately released after transmission (RAT) or released after one more reception (RAR) at the sending station. For efficiency reasons we focus on RAT. In our model, the pre-arrival think time β_1 plus the token-presence time $\beta_2 < \beta_1$ must be smaller than the ring cycle time. In case that the floor is taken, $\beta_2 \approx (\delta + \tau + \gamma)$.

Theorem 5 *The efficacy of STR without multicast support is*

$$\eta_{STR} = \frac{\delta(1 - e^{-\lambda(\beta_1+\beta_2)})}{\frac{n}{2}(\tau + \gamma + \epsilon + \beta_2) + \delta(1 - e^{-\lambda(\beta_1+\beta_2)}) + \iota + \frac{1}{\lambda}} \tag{3.13}$$

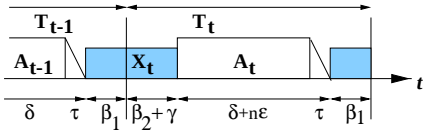

Figure 3.12: Prototypical STR timeline.

Proof: The timeline for STR is depicted in Figure 3.12. We assume perfect floor capture and only one node is active at any time. The average utilization is $\bar{U} = \delta P_s$, with P_s as the success probability that the token is available in the period $\bar{X} = \beta_1 + \beta_2$. The probability that a floor can be claimed by a user is $P_s = 1 - e^{-\lambda(\beta_1+\beta_2)}$. The token cycle time is a function of the active node set n, and with growing session size it is less likely that all nodes will engage in floor contention. The cost to transfer a floor token in a cycle involves on the average $\frac{n}{2}$ nodes, amounting to $\frac{n}{2}(\gamma + \tau + \epsilon + \beta_2)$ including processing overhead $n\epsilon$. The

average turn lasts hence $\bar{T} = \frac{n}{2}(\gamma + \tau + \epsilon + \beta_2) + \bar{A} + \bar{I}$. The idle time is again $\bar{I} = \iota + \frac{1}{\lambda}$. Substituting $\bar{U}$ and $\bar{T}$ into Eq. 3.4 yields Eq. 3.13. The overhead to maintain the token is not included in this result. □

Corollary 5 *The efficacy of STR with multicast support is*

$$\eta_{STR}^{MC} = \frac{\delta(1 - e^{-\lambda(\beta_1+\beta_2)})}{\frac{n}{2}(\tau + \gamma + \beta_2) + \delta(1 - e^{-\lambda(\beta_1+\beta_2)}) + \iota + \frac{1}{\lambda}} \tag{3.14}$$

With multicasting, a token is sent only once to the network interface, but it takes on the average $\frac{n}{2}$ hops to cycle back for another turn option, however, the processing overhead ϵ vanishes.

Tree-based Coordination (STT) is a hybrid control solution based on the idea to perform floor control across a logical tree structure, which allows for more efficient mixing of individual media sources [226] and close correlation of the control geometry with the actual underlying multicast routing tree, or the end-to-end reliable multicast tree. Hierarchical organization of stations also supports inter-group collaboration, subgroup addressing, and allows more scalable and economic transmission of session data [137]. A tree-based group coordination protocol representing STT is discussed in section 3.5.

Control messages in a STT protocol are passed along branches of the tree in a parent-child relation reflecting multicast group membership. No specific node alone is burdened with the obligation to make floor allocation decisions, and tokens can wander freely across the tree branches, without being cast into a specific traversal order other than what multicast group membership expresses. Control messages are aggregated across the tree by coalescing multiple messages of the same type, such as floor requests, into single directives on their path to receivers. Such aggregated management of control information liberates the FH from *control implosion*, where handling of control messages is concentrated in a single node, and allows instead for message exchange in local groups. A control tree is assumed to have a branching factor of K, indicating the number of children for each node. The communication delay depends on the height of the control tree, not the session size.

The average messaging cost is $O(log_{K-1}(n))$. A special case of STT, which equals a moderated session model, a radiating star-topology with the FC at the core, is surrounded by $n-1$ neighbor nodes. This model is efficient for small sessions, however, the FC node may become overloaded or fail for large sessions.

Theorem 6 *The efficacy of STT without multicast support is*

$$\eta_{STT} = \frac{\delta}{(K+1)\bar{P}(\gamma+\tau+\epsilon) + (\delta+\bar{P}\tau) + \iota + \frac{1}{\lambda}} \tag{3.15}$$

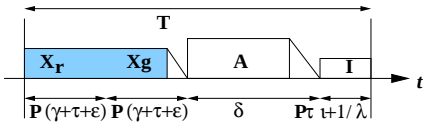

Figure 3.13: Prototypical STT timeline.

Proof: The timeline for STT is depicted in Figure 3.13. We have again perfect floor capture due to explicit token exchange, with $\bar{U} = n\delta P_s$, and $P_s = \frac{1}{n}$. With the normalized average path length $\bar{P}$, the average duration of the floor capture period amounts to $2\bar{P}(\gamma + \tau + \epsilon)$. Assuming that a node does not know the location of FH or FC, exploratory unicast of a control message from one node to its parent and to each of its children would amount to $\bar{X} \leq (K+1)\bar{P}(\gamma+\tau+\epsilon)$ plus a targeted reply costing $\bar{P}(\gamma+\tau+\epsilon)$. However, as outlined in section 3.5, nodes can be tagged with unique prefix-labels, which allow for efficient absolute or relative routing of control directives toward the FC or FH. Consequently, control directive must be sent only to one neighbor node as the gateway on the path to FH, hence $K = 0$. Signaling the conclusion of the activity period adds another propagation delay, $\bar{A} = \delta + \bar{P}\tau$. The activity period may be trailed by another idle period of average length $\bar{I} = \iota + \frac{1}{\lambda}$. Substituting into Eq. 3.4 gives Eq. 3.15. □

Corollary 6 *The efficacy of STT with multicast support is*

$$\eta_{STT}^{MC} = \frac{\delta}{\delta + 2\gamma + 3\tau + \iota + \frac{1}{\lambda}} \tag{3.16}$$

With multicasting, a request-grant pair takes two γ and τ, plus another τ to signal completion of the turn. A close correlation between the control tree of the STT protocol and the end-to-end multicast tree is assumed. A station sends only one message to the network interface, i.e., $\bar{P} = 1$, $K = 0$, and ϵ becomes negligible.

Two other SFC approaches are part of the taxonomy. In *polling*, stations are systematically or randomly probed for pending floor requests by a central station, with polls acting as floor tokens. Delay between polling a secondary node, waiting for its response, and shifting to the next node can be large. Another disadvantage is the reliance on a central coordinator node. In a *reservation* system, floor allocation is divided into a reservation interval and a data interval. The reservation period is finite, but allows for variance in holding duration. However, the natural flow of interaction can not be accommodated by preset reservation periods, which may quickly become obsolete. Polling and reservation were excluded from our analysis, because, to our knowledge, no telecollaboration systems have been built based on these principles. For convenience, the efficacy η^{MC} for the discussed floor control paradigms with multicast support is summarized in Table 3.4.

Protocol	η^{MC}
RSI	$\lambda\delta e^{-2\lambda\delta}$
RSF	$\frac{\delta}{\delta+e^{2\lambda\gamma'}\left[\frac{e^{\lambda\gamma'}-1-\lambda\gamma'}{\lambda\gamma'(1-e^{-\lambda\gamma'})}+\gamma'+\tau+\iota+\frac{1}{\lambda}\right]}$
RAS	$\frac{\delta}{\delta+\tau+\frac{1}{\lambda}+e^{\lambda\tau}(\gamma'+2\tau+\iota)}$
STD	$\frac{\delta}{\delta+3(\gamma+\tau)+(n-1)\epsilon+\iota+\frac{1}{\lambda}}$
STR	$\frac{\delta(1-e^{-\lambda(\beta_1+\beta_2)})}{\frac{n}{2}(\tau+\gamma+\beta_2)+\delta(1-e^{-\lambda(\beta_1+\beta_2)})+\iota+\frac{1}{\lambda}}$
STT	$\frac{\delta}{\delta+2\gamma+3\tau+\iota+\frac{1}{\lambda}}$

Table 3.4: Efficacy of floor control protocols with multicast support.

Results

We contrast the efficacy of the discussed schemes with four cases: (1) a small group in a network with low link latency; (2) a small group in a high-latency network; (3) a large group in a low-latency network; and (4) a large group in a high-latency network. We set $\tau = 0.005\ s$ to characterize a short propagation delay as it is typical for local area networks, and $\tau = 0.4\ s$ for wide area networks and Internet collaboration. The latter value is also an upper bound for acceptable delay in applications dependent on timing relations among separate media streams [212]. Small sessions are set to $n = 5$, which is representative for PC-conferencing systems, and the value $n = 300$ for very large sessions corresponds to traces from MBone sessions [140], sampled over a period of several hundred hours and indicating a group size ranging from 150 to over 550 participants. The time to sense and react to floor information in RSF is much slower than for machine driven sensing, hence we assume a lower bound of $\gamma' = 0.25\ s$. For automatic detection or processing of a control packet of 25 bytes length we assume $\gamma = \epsilon = 0.02\ s$. The token-ring "think times" for arriving and offered tokens are relative to the activity time, and set to $\beta_1 = \frac{\delta}{2}\ s$ and $\beta_2 = \frac{\delta}{10}\ s$. The typical idle time is chosen to be $\iota = \frac{\delta}{5}\ s$. The normalized activity time is $\delta = 1$. Figures 3.15 and 3.14 plots the resulting efficacy for these scenarios.

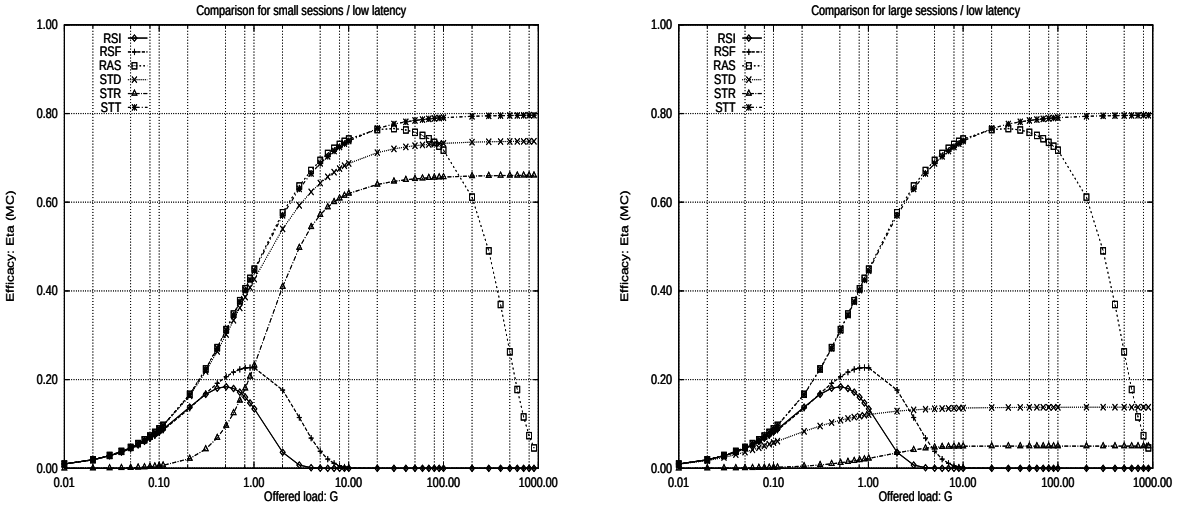

Figure 3.14: Efficacy with multicast support for low network latency.

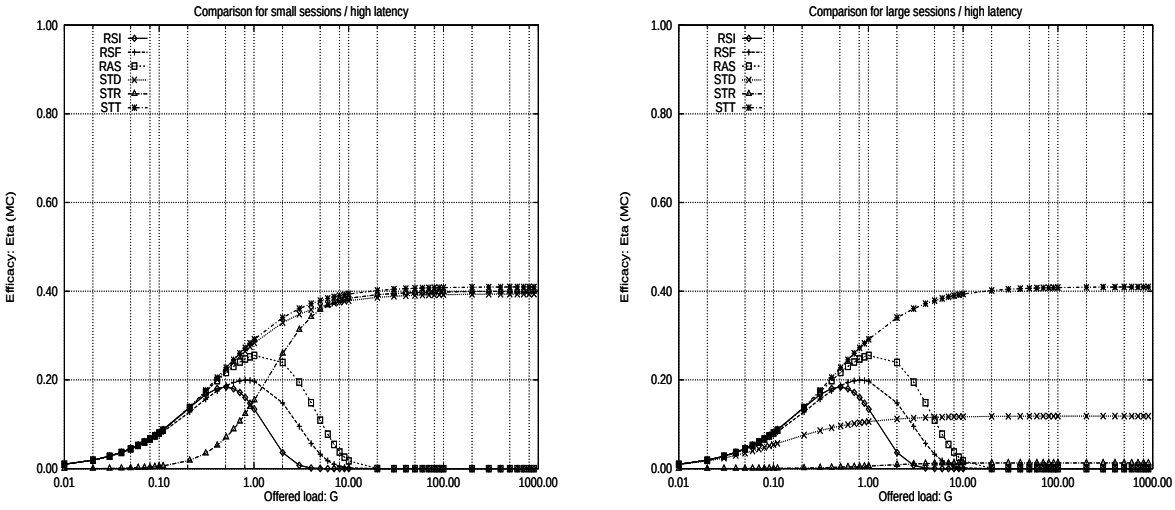

Figure 3.15: Efficacy with multicast support for high network latency.

The efficacy of RSI is below 20% in all four scenarios. Systems employing RSF benefit from coordination attempts and improve slightly over RSI, approximating 25% efficacy in networks with faster links. Both schemes are unstable even at low request rates, Collaboratory efficacy quickly decreases and the average delay becomes unbounded. This models the phenomenon that users avoid frequent turn-taking in telecollaboration [115]. RAS was a first attempt toward unintrusive machine-assisted floor control and performs very well for local area networks, where the system's responsiveness warrants quick updates and steady resource tapping. For high link latencies it allows for slightly higher loads than RSF, however, it also degenerates quickly and rises barely above 20%. STD performs well for small groups, where a station needs only to send few packets to the session remainder, but despite stability for higher loads it barely exceeds 15% efficacy in larger sessions. In its best case of small sessions and low network latency, STR approximates 65% efficacy and it is generally stable, but degrades with rising scale and delay. In particular, it collapses for large sessions with high latency. STT shows the best overall behavior, both in terms of scalability and stability. It reaches up to 80% efficacy in low latency networks, and about 40% efficacy for high link latencies, independent of session size. According to these results, STT fills a gap and is particularly well-suited for Internet collaboration in large groups.

In the following sections, we describe two protocols to implement floor control in fully-connected networks, and in shared multicast trees.

3.4 Floor Control Protocol (FCP)

In this section, we discuss the Floor Control Protocol (FCP), which deploys floor control in direct-link networks without specific assumptions on the node topology. FCP supports queuing of requests using floor passing or floor asking, using various service policies. The design concept behind FCP aims at versatility and adaptiveness to allow usage of the protocol for various resource and session types. The protocol operates distributedly and is assumed to run independently at each host participating in a collaborative session, interfacing with a session orchestration service. Contrary to other control mechanism such as token-passing or activity sensing, FCP allows for concurrent floors, and multiple floors per resource, and does not presume a predefined hop sequence or rely on a short propagation delay between nodes. Communication among nodes is assumed to be solely via message passing, and not in shared memory. Links may be unreliable for data transmissions, but are assumed to be reliable for control message dissemination. We do not presume a multicast transmission model for FCP, however, message overhead will decrease if multicast is used. FCP uses two system roles, floor holder (FH) and floor coordinator (FC) to manifest distributed control, which may be unified or separately addressed in the control process exercised by FCP. The roles depend on the nature of the shared resource, which is either ubiquitous and hence available at multiple sites (e.g., files), localized (e.g., an instrument, which is attached to one site, but whose functions can be controlled remotely), or mediating between several hosts (e.g., an audio-channel).

3.4.1 Data Structures

We define the control roles in FCP as a triplet $\mathcal{CR}= \{FO, FC, FH\}$. The floor owner FO is unique for each shared resource and introduces, owns, and withdraws it in the collaborative workspace. The floor coordinator FC, one per resource, is the principal process regulating

who may attain which floor at what time. The floor holder FH is the current temporary user of the resource governed by that floor. At the beginning of a session, floor ownership and holdership are decided depending on the joining order of nodes to the session. For a resource R_j introduced first by node N_X, this node is by default FO_j, FC_j, and FH_j, until distributed control shifts these positions in the course of collaboration. There may be several floor holders per resource, if the usage semantics of that resource allows for multiple users under mutual agreement, e.g., in the case of multiple graphical cursors in a shared editing system. Information on roles is transferred with each control packet, i.e., each active site is informed at all times about the current overall control state of the collaborative system. Hosts, users, and resources are assumed to be uniquely identified within a session. FCP interfaces with a session directory that keeps track of the current organization of the session, services requests, and tracks floors for local resources and remote operations.

A *control state table* for all floors is kept at each site participating in a session. Session events such as withdrawing, joining, or side-activities in coteries are also reflected in state tables. The local and remote allocation of such tables, their replication and cross-referencing is left to a detailed floor control protocol working in conjunction with session control. The control packet structure to uniquely identify and transmit actual control parameter instances within sessions is given in Table 3.5 and allows for coordinated session conduction and recovery.

Field	Description
SId	Unique session id
HN	Hostname
GId	Unique group id
UId	User (agent) id
RP	Role-based permissions
AId	Application id
RT	Resource/media type and instance
FId	Unique floor id
F#	Number of allowed instances
FS	Current floor state

Table 3.5: Floor control packet structure.

SId and HN are unique designators for the particular session and machine. GId reflects the aggregation of users within the set of running sessions, their relation to groups and multiple activities. UId and RP identify a particular user, his or her function and authentication within the session, comprising both the assigned task (note-taker, speaker etc.) and current status with regard to floor usage (floor controller, floor holder, or participant). A priority value can be attached to allow for preemptive task completion or preferred floor attribution for specific holders. AId identifies the application in use, from which a particular RT, i.e., instance of some resource such as text, voice, video etc., emanates. FId and F# characterize the specific floor in use and its number of concurrent instances. Their values depend on the number of floors that a resource concurrently allows, as in the case of a whiteboard with multiple cursors and telepointers. Finally, FS reflects the actual state of the floor-control protocol, i.e., whether the respective floor is free, granted, requested or in migration. Such states must be tracked for local and remote actions in order to capture and control collaborative events to and from the local site. For time-critical media the control packets needs to be transmitted on behalf of a real-time transport protocol [44, 200].

3.4.2 Operation

The CM running FCP is assumed to be implemented as a middleware component below the application layer, and acts as intermediary daemon process between users and applications. FCP controls concurrent floors in multiple threads of the same state machine. We assume that CMs process input events reliably and on a first-come-first-served basis. State changes in FCP are evoked with floor control messages that do not affect the data packet structures. The floor control algorithm in FCP does not presume a prescribed topology. The protocol states, transitory actions and events are traced in as many instances as there are floors to track. Control state information is disseminated between nodes only when they interact. Each node keeps a local and partially replicated state table, which records the current state of the floor distribution. Table management is weakly consistent, receiving updates for a specific floor only if its node is active. Control packets contain per-turn information on

the sender and receiver nodes, session, user, and floor state, which comprises the resource, activity, and η. FCP differs from prior solutions in its distinction between local and remote control, reflecting the symmetric turn-taking scheme described earlier, which allows for separation of local and remote affairs at each node and tracking of multiple activities on the same resource type among several partitions of the node set. The protocol consists of five main states (LHF, RHF, IDLE, LRF, RRF) to represent submission of floor requests, attempts to claim a floor, actual floor usage, idleness and release, both for the floor holder FH and floor coordinator FC. The control operations executed by FH are `attempt, acquire, use, release, withdraw`, and FC executes `elect, defer, grant`, and `revoke`.

Coordination Policies

FCP works under separation of FC and FH, and with $FC = FH$, which implements a roving floor controller policy ("control follows holder"). This model compares to a hybrid approach between centralization and a fully distributed approach. Control for a specific floor is here always centralized in the node currently holding the floor. Once a turn is completed, control shifts to the next holder. This approach compares to electing a new moderator at every turn and performs favorably for interaction across long-distances, since messages must only be sent from users to one FH, rather than using an intermediary FC. If $FC \neq FH$, a user must send a request message to FC, FC responds with a deny or grant message, eventually transfers the floor to FH, and may send an update to the group to inform all nodes about the new floor holder, which requires three messages on the average. This model can be implemented in two possible ways: (1) a user sends one unicast request message to FH knowing its location, and FH responds with a unicast to either deny or grant the floor, broadcasting (or multicasting) the update to the group interested in the resource; (2) a user send a broadcast (or multicast) to the group asking for the floor, and receive the floor from the current FH via unicast. In both cases, two messages are necessary.

Using $FC = FH$, assume for instance that the current floor holder is located in London, and the next floor holder sits in New York. Collaboration partners in Europe will benefit from close collocation with the London-based FH, and collaborators in the US will experience less delay when obtaining the floor from New York. Only in the case of strong oscillation of floor holders across long distances will the benefit of collocated coordinator and holdership decrease. In addition, this model achieves better scalability and resilience, because floors will be handled by different nodes in the network. We assume that data are partially replicated, weakly consistent, and allow for independent activities and views. For resilience, each node caches its local activities, and identifies a floor proxy, in the case that other nodes are unable to connect to it.

State Diagram

Figure 3.16 shows a specification of FCP in the form of a state machine, with σ denoting the set of states, and α denoting the set of activities and events causing state changes.

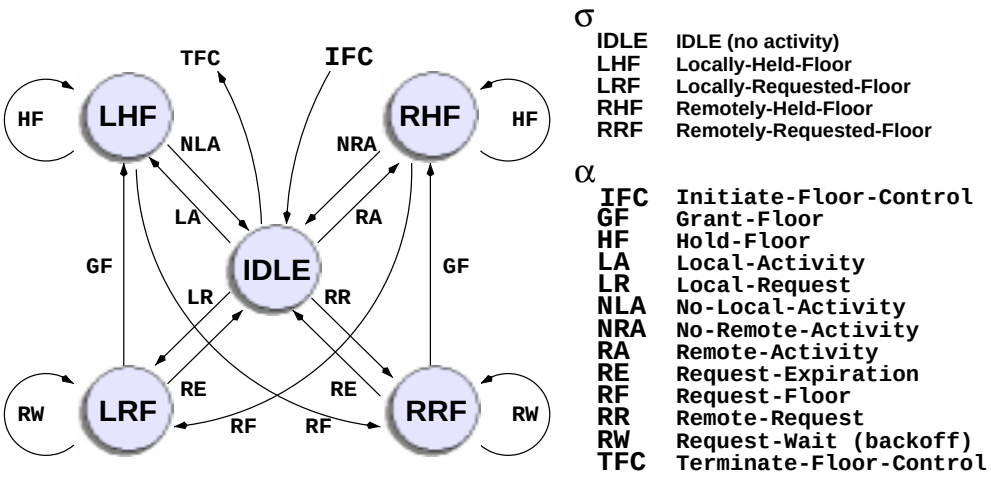

Figure 3.16: FCP protocol state diagram.

The state diagram is simplified in that it does not contain transitions for feedback floors or exceptions such as node failures, which necessitates floor recreation and role elections among the remaining nodes. We assume that sharing for each resource can be switched on or off by the local owner of a resource. The following description outlines the management

procedure, as it is exercised per floor. Once a resource is declared public, FCP by default assumes no activity (IFC into IDLE). FCP remains in the IDLE state, as long as CM detects no messages from either local or remote users, indicating requests to the local FH or FC. Either the local or remote node issues a floor request (LR, RF), inciting a transition into a pending state (LRF, RRF). This wait state serves to sort out simultaneous requests arriving from various nodes according to priority and timestamps or sequence numbers. Nodes can wait for a specified time (RW), after which the request expires (RE) or the floor is granted (GF), using a hold-down timer. A variation of this "countdown-to-election" scheme is to acknowledge requests individually and serve them based on sequence number information tagged to a request. Acknowledgments on floor requests are more efficient in comparison to using a hold-down-timer, if network delays vary.

If the local node acquires the floor (LHF), information produced in accessing the shared resource SR is sent to the local application and disseminated to remote CMs. Local operations and data affecting the resource may have been buffered and can be applied in batch-mode. If the protocol enters the remote state (RHF), the local CM disallows local input to affect SR. Information received from remote on resource updates is percolated up to the affected applications. The turn duration can be based on a timer, incoming preemptive requests, or the user releasing the floor. Once the local or remote node acquires the floor as FH and uses the resource (LHF, RHF), it can continue to do so (HF) until a floor request from a remote node is registered (RF), shifting the protocol to a wait state for the other node. All request, wait, and hold actions are timed, or can be revoked preemptively by FC and observe the turn-taking rules described. The time interval that FC takes to decide upon the succeeding floor holder must be large enough to allow every other potential FH to send requests to this site. Enforcing acknowledgment on floor migration implements atomicity of the floor state change. As long as an idle floor is claimed or in migration to another node, it is attributed to the previous FH. If a migration fails, the previous FH is the default recovery location.

While FCP does not presume a specific logical topology among nodes, an underlying multicast protocol [237] may impose a specific delivery strategy on the sending and receiving nodes in a session. Resource usage is, depending on the floors granted to other sites for that resource, multicast to all members of the multicast group collaborating on the resource. In case of simultaneous claims for the same floor, the race condition is resolved by the request packet arrival order and floor sequence index, or permuted based on an agreed-upon floor policy. State transitions and timers are attuned to specific media characteristics. Multiple incoming floor requests may or may not be queued, depending on the resource and service policy. Without queuing, unsuccessful users must retry until they get the desired floor. In order to render floor management unobtrusive and integrated with operations on shared resources, control must be anchored within the resource handling semantics, e.g., in the form of voice-activated floor allocation for audio conferencing. Temporary withdrawal or leaving of a session terminates floor control (TFC) for the resources from the leaving host, with the option to restore the previous state in the case of rejoining from a log file that FCP maintains throughout a session. To minimize waiting time on floor requests, a RF discard strategy can be implemented with limited queuing, if too many RFs are received at a floor server. Fairness of FCP is resource-specific and depends on the established service policy.

3.4.3 Correctness and Fairness

A floor control protocol can be validated in its correctness, that is safety and liveness [25] and fairness. Safety means that the protocol assigns only one floor per turn to a user, and that the floor being assigned is the one requested. Liveness means that the protocol never enters deadlock and that no user starves by waiting indefinitely to attain a floor. Fairness is inherently given with the established floor policy. We show that FCP guarantees safety and liveness, even for the generalized case that $FC \neq FH$ and K floors being attainable to control the same resource type, such as telepointers. More specifically, if a resource allows K concurrent floor holders, FCP permits at most K nodes to access the resource at any time,

and every floor request is serviced within finite time. We assume failure-free communication in the network and a bounded end-to-end delay between nodes. The following arguments assume a unique and totally ordered indexing scheme among the nodes and activities, atomicity in the transitions between protocol states, and that no code segment associated with any event in the protocol can delay execution.

Theorem: FCP *is safe.*

Proof: We have to show that FCP does not grant resource to two or more parties at the same time. Assume by contradiction, that for K nodes being granted a floor, $M > K$ nodes actually have the floor. Floors F_{ji} for a resource R_j are indexed with labels $i = 1, \ldots, M$, defining an order $(F_{j1}, N_1) \leq (F_{jK}, N_K) \leq (F_{jK+1}, N_{K+1}) \leq (F_{jM}, N_M)$. In order to attain a floor, node N_{K+1} must have received a GF message from the current FC_j. There are only three possible cases in which FC_j may have received a request message RF. (1) A node $N_X, X \in 1, \ldots K$, was already attempting to gain the floor from FC_j with floor sequence number F_{jX}; in this case, FC_j would then have deferred the request from N_{K+1}. (2) Node N_K was already operating on the resource in a previous turn; in this case, FC_j would not grant another floor, until one of the nodes $N_1, \ldots, N_K$ releases at least one floor instance, since $F_{jK+1} > F_{jK}$. (3) At least one of the nodes $N_X, X \in 1, \ldots, K$, is IDLE such that its floor is revoked after a timeout; in this case, request F_{jK+1} is mapped onto F_{jX} and hence node N_{K+1} can never attain a floor, which contradicts the assumption. □

Theorem: FCP *is live.*

Proof: We have to show that (a) no deadlocks occur, and (b) no livelocks (starvations) occur. Ad (a): A deadlock happens when fewer than K nodes hold a floor and a RF message from node N_X is unserviced. Suppose that a deadlock at turn T occurred and the collaboration is deadlocked. Using the previously employed indexing scheme on floors, a RF can be deferred indefinitely only if either FC_j or FH_j defer to reply to N_X. If FC_j does not respond within timeout, a new controller FC'_j for R_j is elected. If the current floor holder FH_j, with $1 \leq \ldots \leq FH_j \leq \ldots \leq X \leq \ldots \leq K$, does not reply, the floor is revoked from FH_j after a timeout and is assigned to node N_X. Therefore, at least one of the $N-K$ nodes

that does not currently hold a floor, eventually receives the floor, which is a contradiction. Ad (b): A node is said to starve, when its RF message is indefinitely deferred while other nodes are being served. Consider node N_X trying to attain floor F_j by sending a RF to FC_j. This request may be immediately serviced, or deferred if another node N_Y holds this floor, or is attempting to do so with a lower request index $h < j$. At most $j - h$ other requests may be served before R_j is served; therefore, $N_X = FH'_j$ eventually acquires the floor F_{jX}. □

3.5 Hierarchical Group Coordination Protocol (HGCP)

3.5.1 Multisite Group Coordination

We outline the operation of the Hierarchical Group Coordination Protocol (HGCP) as an example for STT protocols. Although tree protocols have been the subject of related research, e.g., in mutual exclusion [182], channel access [25], or reliable multicast [137, 173, 237], to our knowledge no tree-based group coordination protocol has been proposed in the literature. HGCP represents two innovations: floor control is inherently hierarchical, and the tree-based mechanism for propagating group coordination is assumed to be closely correlated in its operation with an underlying tree-based multicast service. The underlying multicast tree is assumed to be shared by multiple sources. The CBT protocol [19] builds such a shared bidirectional tree for multicast routing, centered around rendezvous points called cores. The LORAX protocol [137] builds a shared end-to-end tree used for error recovery in reliable multicast. The HGCP control tree mirrors such a shared tree, unifying forwarding, reliable transport and coordination in one coherent infrastructure.

LORAX [137] uses a single shared acknowledgment (ack) tree per session for multiple sources, instead of creating a separate tree per transmission. The LORAX ack tree is characterized by labeling of nodes from a finite alphabet, with the property that the label $l(x)$ of a node x is the prefix of its children. The purpose of these labels is to provide relative addressability of multicast groups and subgroups. The maximal length of labels in a tree

reflects its depth, and labels in the tree remain constant for the lifetime of the session, except in cases when nodes withdraw or join. Adding a node in the tree involves only the new node as a child and its parent, while deletions require relabeling of the subtree of the deleted node. The label cardinality depends on the tree branching factor and the session size. A tree with n session participants and branching factor K has $log_K n$ levels, with $log_2 n$ bits needed for node labels. The ack tree does not require modification to the router-internal IP multicast infrastructure. However, Levine and Garcia-Luna-Aceves have generalized labeling to the router level [136], exploiting novel mechanisms of streaming and addressing. The ack tree structure enforces cascaded acks and negative acknowledgments across tree branches, and prevents receivers from contacting the source directly to confirm reception of packets or ask for retransmission of lost packets. Instead, recovery is contained in local groups of tree subbranches by aggregating ack information.

In HGCP, the ack tree becomes the control tree of a *CE* session and shrinks or expands dynamically during session lifetime. Stations are prevented by HGCP from contacting the FH directly, and instead submit control directives for aggregation in their immediate tree neighborhood, achieving better scalability in controlling the floor for large sessions and many resources. HGCP derives label information from the end-to-end multicast protocol to administer floor information and route control directives between hosts contending for a resource. HGCP is hybrid in that FH poses a centralized, but roving point of control for an individual floor; however, the protocol runs in every node starting or joining a session and the union of distributed floor states in every active node yields the global control state.

3.5.2 Data Structures

Control responsibilities are distributed over the entire session tree, dividing the session into local groups with three kinds of nodes: a control node, relay nodes, and leaf nodes. The *control node* hosts the FH (or FC), regulating access to a resource R and transmitting updates concerning R. *Relay nodes* collect coordination (control) directives (CDs) from their children and forward them in the tree towards the FH. Likewise, they relay replies

back to their children. A relay node, which is not a member of the destination multicast group for a directive is called an extra node. An extra node can be viewed as a proxy for the destination node. *Leaf nodes* delimit tree branches and communicate solely with parent relay nodes.

For messaging efficiency, HGCP unifies the roles of FH and FC and locates them jointly in one node. If both roles would be separated, three nodes are involved in a triangle communication: a node x would first have to contact the FC, who would await release of the floor from the current FH, and then grant the floor to x by asking FH to transfer the floor to x. With unification of FH and FC, a moderated session style can still be supported, by sending the floor back to the moderator node after turn completion for assignment to the next floor holder. Role unification is also practical for loosely coupled sessions, which need no distinct leader and are destined to make progress in self-moderation. FH does not have to know the identity of session members, because generic labels provide sufficient routing information, i.e., HGCP supports anonymous collaboration.

A coordination directive CD consists of a list of source labels $\{l_s\}$ for the aggregated delivery of directives from multiple nodes, a list of destination labels $\{l_d\}$, a timestamp (or sequence number) #, a time-to-live field TTL, an identity descriptor ID for public, private, or anonymous submission of directives, and a floor descriptor f containing floor information as specified in Chapter 2:

$$\text{CD}(\{l_s\},\ \{l_d\},\ \#,\ \text{TTL},\ \text{ID},\ \text{f}) = < \textit{directive} >$$

The TTL field sets an expiration date on persistent CDs. Setting TTL = 0 configures nonpersistent floor allocation. Values for $< \textit{directive} >$ are: `REQUEST` to ask for access to a resource; `GRANT` to grant permission to access the resource, provided that the floor is free; `DENY` to signal that the floor is taken or reserved; `RELEASE` to relinquish the floor; and `STATE` to retrieve updated floor state information. Each host in a session is client and server for coordination directives (CD) to other hosts. CDs are issued between hosts to synchronize their joint tasks, to implement causal or total ordering in distributed events, and to mitigate access to shared, but exclusive resources.

Each CD contains the source's label, the target label(s), a sequence number, a local timestamp, a session-wide unique resource id obtained from the session directory, and a floor id, which denotes the temporary access permission or an activity descriptor for a resource. The structure of a CD packet is shown in Figure 3.17. **V** denotes the version. **CDid** identifies the coordination directive. **Typ** denotes the type of operation, characterizing various resource modalities. **TTL** indicates the scope of the CD. **Opt** is reserved for priority codings. The timer and sequence numbers tag the CD uniquely in the session and event space. **Checks** is the checksum field. The **Source addr** and **Target addrs** fields contain the labels for the sender of a CD and its target nodes.

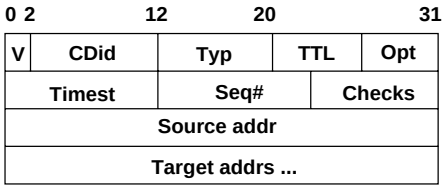

Figure 3.17: Packet header fields for coordination directives (CDs).

3.5.3 Operation

HGCP can implement nonpersistent floor control, with any request except the first accepted one being discarded, or persistent floor control, where a finite request queue is maintained to keep track of consecutive requests. It regulates messages exchanges between peer user agents, which are either users or applications acting on behalf of users, with the goal to intercept data and commands between users and ensure that only one user can access a particular resource at any given time. Each user agent tracks one or more floors, one per resource. The floor state is controlled in a distributed fashion. Call setup, late joining and withdrawal from a session are handled by a membership protocol interfacing with HGCP.

HGCP scales well because nodes only maintain a local picture of their immediate session neighborhood, knowing the direction toward the current FH for a specific floor, and communicate only within the local group defined by parent and children nodes. FHs are positioned at the root of the tree, and the tree is virtually rotated at floor hand-over towards

the new FH, whereby balancing properties of the tree may change. Less interactive stations are more likely placed on leaf positions. On the average, leaf nodes must compare more bits in the control routing procedure than nodes close to the root. Compared to a geometry in which nodes are fully connected, trees significantly reduce the amount of messaging. Labels also allow for communication between distinct coteries of hosts, without involving the entire multicast group.

The *setup phase* of HGCP can be initiated along with or after session advertisement via a session directory service. If the session is open, new nodes can join, otherwise invited stations are only allowed to withdraw. Each station maintains a floor state table containing floor information about the local and remote resources being part of the shared workspace. The state table is compared at regular intervals with the tables of parent and children nodes in the tree, and is incrementally updated, if a younger table is detected. In this sense, floor agents are a cooperative collective, building a consistent distributed control state for the entire session. Every station advertises its shared resources and floors to neighbor stations in the reliable multicast tree and integrates updates from these stations into its local floor state table. The station introducing a resource R to a session S creates and injects the floor token for R into S.

The *active phase* of HGCP concerns the aggregation and forwarding mechanism between local groups in the control tree for the various node types. A node can access and use a shared resource R only if it becomes FH. The floor for R is idle if no station sends messages to the network concerning R. To acquire the floor, a station sends a `REQUEST` CD to its neighbor station in the control tree, which is closer on the path to FH. The direction is computed by taking the prefix of the location entry for the floor marked in the floor state table. Each request is compared against requests pending from other nodes. According to the aggregation semantics, if a near-by node is able to provide a CD response, the CD request is served by this node and not propagated to the FH. Compounded CDs received at FH are ordered and serviced first based on priority, second with regard to queuing, timestamp and reception order. If the FH is static, eligibility for the floor may also be

decided based on statistics, indicating frequency and holding time per floor for each node to induce fairness into the floor acquisition process. When FH completes its resource access, it sends a `GRANT` CD to its successor, based on the source label information. After receiving confirmation, FH officially relinquishes the floor by multicasting the position of the new floor holder, denoted as FH', to the session. A new turn cycle begins, with nodes contending for the floor now submitting `REQUEST` CDs to FH', which may also receive a copy of the request queue and hand the floor to the next station in the queue when finished.

In the *termination phase*, floor state information is deleted from the records of the local station, but retained in a log file for turn-taking history. Single nodes can withdraw intentionally or by failure from a session. In normal operation, any node not interacting and affecting the global floor state can withdraw. Consider the case that a station that submitted a request for the floor withdraws before a `GRANT` CD reaches it. A successor station FH' must confirm reception of the floor before the remainder session updates its state table on the location of FH'. Hence, pending but unconfirmed floor hand-overs are never registered. If FH wants to withdraw from the session, HGCP revokes the floor from this node and assigns it to the node with the oldest pending request. If a relay node withdraws from the session, its children and parent link together and re-initiate any operation pending on the missing relay nodes.

Hosts need to maintain locally the following state: the resource ids shared from local to the remote hosts, and the remote resource ids accessed locally, together with their CDids; a state table indicating which resources are locally available or held remotely, and the id of the remote FH; and a request queue $ReqQ$ which collects successive CDs from different hosts (the queue is limited by the number of nodes in the session). If a hop node receives the same CDs from different nodes, it aggregates them into one CD and checks to determine if a response to these requests can be satisfied locally by polling its own state and the state of neighboring nodes. Otherwise the composite CD is self-routed up or down in the tree toward the target node(s). This implies that the number of CDs required to coordinate nodes decreases as the request activity increases, because requests are not sent further, if

a hop node is reached that has already processed the CD. The target address may also contain the name of a multicast group, which is then resolved into its members locally at the primary receiver node for this group. A joining node retrieves the current control state for resources of concern by polling its parent node.

In addition, each node maintains a FIFO queue of pending CDids, identified by the senders' labels. A hop node receiving a floor compares the label of the elected node with the head of the queue, and self-elects if its own label matches the head, or forwards it in the routing procedure outlined above. A control node receiving a request responds immediately to the request by sending back a grant message to the requester, if its local queue is empty, or it appends it to its queue. When control shifts from a node to another one, the pending request queue is transferred to the new control node and its new address is multicast to all groups sharing the associated resource.

The example in Figure 3.18 illustrates the operation of the protocol in a binary control tree and with a binary labeling alphabet. The protocol stack indicates that HGCP, run by a floor agent FA, is collocated with a session management agent SA, interfacing with reliable multicast below and applications above.

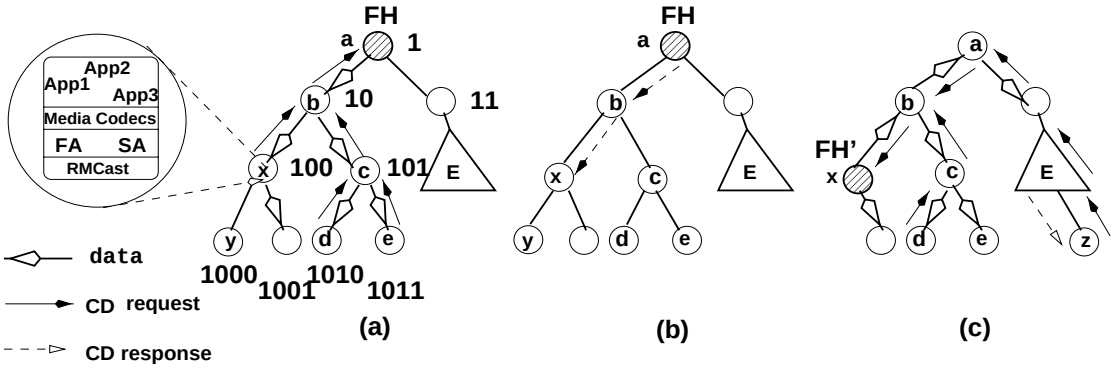

Figure 3.18: Sample HGCP scenario.

In scenario 3.18(a), FH for some resource R is initially placed at the root a, with $l(a) = 1$. All nodes have a consistent update on the current FH for various resources in their floor state table. FH multicasts data updates of its work on R, indicated by hollow arrowheads, to all the sites in the multicast group, which may process the update or discard it. Switching

a remote resource on or off, such as a video stream, compares to a local "receiver floor", by which a station can individually configure its stream and data reception. Floor control exerted by HGCP concerns only "sender floors". Consider the case that nodes x and e contend for the floor of a shared resource R. Looking up the FH entry for the resource in question, they send a `REQUEST` CD to their parent, because $l(FH) = prefix(l(e))$, and $l(FH) = prefix(l(x))$. For the parent node c of e, $l(FH) = prefix(l(c))$, and similarly, $l(FH) = prefix(l(b))$. Hence, both requests cascade upward in the tree toward the root. Node b is the relay node for x and e and either already has forwarded the request from x to a, if CD(e) arrives after CD(x), or it aggregates both CDs into one request, and propagates it to a. Assume that the request from x arrived earlier or that it is prioritized. Once FH finished and released the floor for R, or when its floor expires, it multicasts a `GRANT` CD across b to x. Once x confirms reception of the floor, a multicasts an update with label information $l(FH') = l(x) = 100$ to the remainder of the session, indicating the start of a new turn.

In scenario 3.18(b), x is the new FH', and starts using R and multicasts updates to its children. All nodes in the multicast group propagate their CDs towards the location of FH', as indicated in scenario 3.18(c). Node y dropped out of the session and all its shared resources r_i with $FO(r_i) = y$ are withdrawn and floor states tables across the session are marked up accordingly. A new node z in subtree E joins the session and sends a `STATE` CD to its parent after integration into the control tree, retrieving the current state table for its local resources shared with the session. Assume that the parent node of local multicast group E has the most recent state table (all nodes are steadily querying their neighbors and eventually converge towards the most current state table.) Then the update request from z remains in the right subbranch of the tree and does not need to be forwarded to FH', i.e., target nodes such as FH' are relieved from the obligation to handle every request from within the session. After this update, node z can start contending for the floor with the other active nodes by propagating CDs toward FH' based on label information.

The example shows that labels are valuable both as absolute and relative pathfinding markers in the control process. Reliable multicasting and aggregation of CDs and floor states between a node and at most K children (instead of the entire session) allows for more economic storing and forwarding of control traffic.

3.5.4 Resilience

The shortcoming of a tree model is that a damaged branch affects all children nodes, which may corrupt the coordination process unless it is a leaf or branch that breaks off. Such exceptions must be communicated to other on-tree hosts. We will not discuss repair and maintenance mechanisms for trees, since this is the task of the underlying reliable multicast tree protocol, however, we discuss the impact of breakages on the communication of CDs, control roles, and the consistency of floor information.

Several cases must be considered when nodes or links fail: (1) a node fails, either (a) as FH, or (b) requesting a floor, or (c) being uninvolved in any floor allocation process; and (2) a link fails (a) with coordination directives in transition, or (b) otherwise. In case (1a), if FH fails holding the floor, a timeout and election protocol among neighbor nodes ensures that the lost floor will eventually resurface at one of these neighbor nodes. In cases (1b) and (2a), a pending request may be fulfilled after a station disappeared from the session. Since this station will not confirm the floor transfer within a timeout period, the floor will be reassigned to the next contending node in line. In cases (1c) and (2b), any other node may withdraw from the session any time; however, if this node is an extra node for a floor transfer, the parent and child node embracing this node must first reestablish a connection to be able to conclude the transfer as proxy nodes for the failed node. Every floor state alteration is hence treated as a transaction, and interrupted floor changes are reset to a consistent state before alteration, or committed when succeeding. In the nonpersistent model, nodes must resubmit their failed requests after a timeout.

Previously, we assumed that transfer of CDs and accounting of control information among nodes is failure free. Even we if assume reliable multicasting to ensure that CDs are

eventually transferred, the control apparatus may need additional recovery mechanisms to ensure consistency. This applies to regular node failure, control node failure, link failure, or token loss or duplication. Such exceptions can be preempted by redundant dissemination of status information, or by detection of loss and recovery. Regular node or controller failures are typically detected via timeout and recovered with an election protocol, with neighbor nodes providing state updates. Continuation of a split session is possible if the members in each partition agree to continue, e.g., if a quorum exists.

One method to deal with the case that a CD is lost completely or reaches only a subset of nodes is to multicast a `CD_probe` message from a node i to the session remainder. The response time t_r to receive a response is bound by the maximum time for the probe to traverse the longest link, t_{max}, plus the time t_{ack} for the receiver nodes to send a positive or negative acknowledgment, $t_r = 2t_{max} + t_{ack}$. If the CD is diagnosed as lost, the controller node for the respective floor must regenerate the token and send an update to the session. In the following paragraph, we show that HGCP is safe and live for the case of a static FH.

3.5.5 Correctness and Fairness

Theorem 7 HGCP *is safe, i.e., at most one node receives the floor for a specific resource at any time.*

Proof: A floor control deadlock exists if node x holds the floor and at least one or more nodes request the floor, but are unable to acquire it in finite time. There are several potential reasons: (1) A node operates on the resource without being really granted the floor; if there is no legitimate FH, the floor privilege cannot be passed on; (2) the `GRANT` CD is transmitted but does not reach any requesting node; and (3) FH is unaware of requests. The proof is based on the idea that the control tree is acyclic and propagation of request messages between nodes towards the FH prevents circular stalls of control messages. Using induction on the height of the K-ary control tree, the tree reduces for $h = 1$ to a non-hierarchical star-based scheme with $n = K + 1$ nodes.

In case (1), the floor semantics states that a floor is assigned to one node at all times, that is either FH, or one of the other K nodes in the session. If the node currently holding the FH privilege is different from the node legitimately holding the floor, this node will acquire the FH tag. In case (2), a FH candidate node is either nonexistent or withdraws from a session, while the floor is in transit. As stated earlier, floor transfers must be confirmed by the receiver to the sender, and stalled transfers are nullified. In case (3), if FH is unaware of requests, this node is eventually timed out, and the FH privilege is shifted and assigned to one of the K active neighbor nodes. If a node x, as one of the active nodes, sends a `REQUEST` CD towards the FH, it is received by the FH assuming reliable transmission. Upon reception, FH sets a requested flag and, if queuing is supported, it inserts the label information of the requesting node in its FIFO request queue. Based on the queue semantics, the floor will eventually be shifted to node x.

For $h > 1$, we assume that the theorem holds for any height l with $1 \leq l < h$ and we need only show that the theorem also holds for $l = h$. The mechanism of transferring floor holdership, as discussed in the simpler case, can only be bypassed if a `REQUEST` CD is continuously outrun by a `GRANT` CD. However, the implicit routing scheme based on labels defines a forwarding order on nodes, that defies indefinite stalls of `REQUEST`. No node can indefinitely hold on to a floor, because floors expire or peer nodes will revoke the floor. The FH must eventually become legitimate, or floor transfers must be confirmed, or FH must become aware of other nodes requesting the floor based on the principle that `REQUEST` CDs are propagated steadily towards the FH. Thus, directives toward and from FH must eventually arrive at their destination node, and thus deadlock is impossible. □

Theorem 8 HGCP *is live, i.e., a request by any node x must be served in finite time, and no node suffers from "floor starvation".*

Proof: HGCP is live because the FH node either services the first request and discards all follow-up requests, or it maintains a request queue with nodes marked by prefix labels. This queue cannot be longer than n, given that no node is allowed double entry until other nodes have been served. Each of the possible actions of FH will reduce the request queue

ultimately to length 1, which allows the remaining node to eventually acquire the floor. Hence, every node requesting the floor will be served in finite time. □

Fairness refers to the frequency and duration, by which nodes acquire a privilege on the average for a given period, which is for example the session lifetime, or the lifetime of a shared resource in a session. Network latency, geographic distance and location of nodes, or varying host capabilities can all be factors in causing uneven dissemination patterns of CDs and unfair allocation of floors. Leaf nodes take more time to propagate their requests across the root to a node on the other side of a tree, than nodes just below the root. Shared trees also do not provide shortest paths between a source and its receiver set [112], which may cause increased latency in CD transfer. It is hence important to establish service policies, which counteract these factors. One simple solution, a "least-recently-served" policy, can be enacted by letting each node maintain a local record of the most recent CDs and their originating nodes. Those nodes are serviced first, which do not appear on the list, or appear last in time or frequency of service.

3.6 Discussion

In this chapter we focused on contention resolution with floor control protocols and their analytic comparison, making three contributions. First, a novel taxonomy for floor control protocols has been proposed based on operational principles. Second, a performance analysis in accordance with the taxonomy has been presented, subsuming rather different protocol classes in one coherent framework. This framework merges time aspects of protocol operations with end-user behavior and thus accounts for both internal and external factors regarding control of resource sharing. Although strong assumptions were made to keep our analysis tractable, this is to our knowledge the first attempt to quantitatively characterize the effectiveness of various strategies of floor management in network-centric group collaboration. We found that an approach based on a logical tree structure outperforms other control schemes in terms of scalability and efficacy. Third, a novel floor control protocol operating in a logical shared control tree has been outlined, whose operation is correlated to

a tree-based end-to-end reliable multicast protocol, permitting more practical implementation and inherent adaptation to membership and link changes in the shared multicast tree. Furthermore, such an approach enables important functionality for collaboration of small user groups within multicast groups, such as selective subgroup addressing and subcasting. Future work must address issues of session stability, authentication, message ordering, and fairness. We conclude that tree-based floor control, embedded with reliable multicast, represents a promising approach to support large-group interactivity across local-area and wide-area networks.

Chapter 4
Ordered Multicast

In this chapter we focus on the ordered multicast distribution of coordination information, such as activities, state updates on multimedia resources, and control primitives among session members. IP multicast communication [47] generalizes the point-to-point and broadcast communication model to multipoint dissemination of messages. A source must send a packet only once to the network interface, and packets are transparently replicated on their transmission paths to the receivers. This form of communication is indispensable for networked applications with high-volume data transfer, such as distributed software updates, news casts, video-on-demand, and interactive applications, for example distributed interactive simulations or telecollaboration systems. Data handled by these applications fall into two categories, continuous media streams and non-real-time data. Real-time data delivery, e.g., for video or audio streams, is typically best-effort and unordered, but must observe deadlines to be useful for an application. Non-real-time packets carry discrete data, and may require reliable, ordered delivery based on the application semantics. With IP multicast, no guarantees are given for reliable or order-preserving delivery of packets to a multicast group.

4.1 Motivation

Changes in datagram routing or transmission errors may cause packets to arrive at their destination out-of-sequence. Disordered delivery of packets in a distributed application may result in different views of the group state at end-hosts. Ordering of messages com-

pensates for the lack of a global system state and the effects of asynchrony, unpredictable network delay, and disparities in host processing in distributed communication. It assures that destination processes observe the same order of reception of messages. Ordering is complemented by reliability and atomicity. Reliability guarantees that messages eventually arrive correctly at their destinations, and atomicity guarantees that a message is received by all members of a multicast group or none.

Consider a distributed interactive simulation with many moving, interacting entities, where a message m_1 is reliably multicast from source s_1 to receiver group Rec_1, and m_2 is reliably multicast from s_2 to Rec_2. A host which belongs to Rec_1 may receive message m_1 before m_2, while another host belonging to both groups may receive the messages in the opposite order. Correct operation of the simulation system requires not only that the input stream is equivalent for all replicas, but all input events have to be delivered to replicated instances of shared applications in the same order. Some ordering protocol must intercept, or better, be integrated in the delivery process to guarantee such consistency.

The majority of existing reliable multicast solutions [162] lack ordering services. A comparison of the performance characteristics of such protocols [137], entailing sender- or receiver initiated protocols, ring- or tree-based protocols, and tree protocols with negative acknowledgments and periodic polling, showed that the latter protocol type is the most scalable and efficient approach known to date among deployable systems. Based on these observations, our objective is to examine how ordering services can be integrated with reliable multicasting, in particular with tree-based protocols, preserving scalability and efficiency. In particular, our objective is to find a solution for *symmetric* ordered multicast, where any node in a tree can be sender or receiver, and where multicast groups may overlap without causing duplicate deliveries.

4.2 Related Work

Much work on total and causal ordering for multicast is centered around fault tolerance or consistency issues in distributed systems. Chandra and Toueg [35] have shown that total

order broadcast and consensus are equivalent problems in asynchronous systems. Many protocols from this field suffer from high overhead to achieve fault tolerance by introducing messaging to implement a failure detector and consensus mechanism. Mayer [148] introduces synchronization delay as a measure to analyze the delay behavior of three multicast ordering protocols to achieve ordered delivery. Cheriton and Skeen [39] argue that CATOCS (Causally And Totally Ordered Communication Support) attempts to solve state problems at the communication level in violation of the "end-to-end" argument [192], based on the hypothesis that the provision of ordering services below the application using these services is insufficient, because it cannot capture the application-specific semantics.

Symmetric or decentralized approaches are based on the total order solution by Lamport [128], where timestamps are assigned to messages at broadcast time. The algorithm by Chandra and Toueg [35] executes reliable broadcast in four communication steps with a weak failure detector, and the solution by Dolev *et al.* [50] is based on a majority consensus protocol. Both schemes have $O(n^2)$ message complexity. Newer approaches such as the TO_multicast protocol [121] or Scalatom [187] incur message complexity $O(r^2)$. Hybrid algorithms using distributed messaging in conjunction with centralized sequencing [186] have been proposed to achieve better scalability.

The ISIS system [26] implemented *two-phase ordering* using vector clocks for logical time keeping, an explicit membership model and layered microprotocols such as CBCAST for causal ordering and ABCAST for total ordering. Later versions of ISIS have adopted a ring-based approach.

Centralized reliable broadcast approaches have been proposed in the RBP protocol [37] with a rotating token to identify the sequencer; the AMOEBA system [122], where the kernel passes messages to a dedicated sequencing processor; or the protocol by Navaratnam *et al.* [155], where a primary manager orders messages and forwards them to a secondary manager heading each local multicast group delivering to application processes. Centralized reliable multicast protocols such as URGC [8], MTP [15] or XTP [214] follow a similar principle.

Protocols of the *ring* type, such as TOTEM [13], RMP [233] or TPM [179] implement total ordering and feature resilience toward network partitions and process failures. A related approach is the token-bus protocol MLMO [239], which ensures causal and total ordering in a hierarchical infrastructure to connect internetworks.

The *tree* protocol by Ng [159] implements reliable broadcast with source-ordered delivery to a single group allowing up to k host failures, building a minimum spanning tree between hosts. Total ordering is achieved using two vectors of time-stamps at each host to keep track of the last message sent to and received from each neighbor. A broadcast message with timestamp t is delivered to end processes only if all messages with smaller timestamps have already been received. The MP protocol by Garcia-Molina and Spauster [87] solves single source, multiple source, and multiple group total ordering, including the case that messages are addressed to different, overlapping groups. However, delivery from the last intermediate host to final destinations is unicast, causal ordering is not supported, and reliable failure detection is required for the protocol to operate resiliently. The improvement by Jia [118, 205] clusters hosts into metagroups representing intersections of overlapping groups and forming propagation graphs based on metagroup relations to minimize duplicate deliveries, enhance parallelism in delivery, and shorten propagation graphs. We will incorporate these significant solutions in our taxonomy of ordered multicast solutions, depicted in Figure 4.1 in Section 4.4.

Various tree-based reliable multicast protocols, TMTP [237], RMTP [173], or LORAX [137], have been proposed in recent years. TMTP builds a source-based tree for flow and error control, where participants are organized into domains with a single domain manager responsible for error recovery and local retransmission in each domain. In RMTP, reliability is achieved by leaf receivers in the delivery tree periodically sending status messages to their designated receivers (DRs), which cascade their status reports periodically upward to higher layer DRs, until reaching the sender. Lost packets are recovered by local unicast or multicast retransmissions by their DR. LORAX employs positional labels in

a shared ack tree for concurrent multicast, however, all three protocols lack support for multi-source and multi-group ordering.

Yavatkar and Griffioen [236] discuss the CLIQUE toolkit for tailoring IP multicast group communication services to the requirements of specific applications, and point out the importance of causal ordering, but omit a solution description. Similarly, Aggrawal and Paul [2] propose a multicast conferencing architecture based on heuristic Steiner trees, however, the problem of ordering is not considered. TOM is geared towards adding ordering to reliable concurrent multicast or concast, i.e., the multicast-style transmission from multiple senders to a single receiver [215]. We base our ordering mechanism on LORAX, but it could also be deployed in TMTP with domain managers, and in RMTP with designated receivers as intermediate ordering nodes.

Another issue is the choice of the underlying transport protocol if continuous media are involved. The TENET approach [44] is focused on delivering performance guarantees, is reservation-based and includes RMTP (Real-Time Message Transport Protocol), which is connection oriented with unreliable delivery, and CMTP (Continuous Media Transport Protocol), which supports periodic network traffic. Both protocols run above RTIP (Real-Time Internet Protocol) with guaranteed services based on deterministic and statistical QoS guarantees. IETF proposes RTP [200] (Real-Time Transport Protocol) for multiparty-multimedia conferencing, combined with RTCP (RTP Control Protocol), which conveys information about participants of a conference, as well as media encoding, synchronization, framing, error detection, encryption, timing, and source identification. RTP does also not prevent out-of-order delivery, uses ST-II [213] or UDP/IP, and hence cannot assume that the underlying network is reliable and delivers packets in sequence. It is used for the MBONE tools, such as VAT.

4.3 System Model and Assumptions

Our network model $N = (H, C)$ consists of a set of n hosts H and communication links C, communicating via message passing in the absence of physical clock synchronization. A

host is equated with the processes running on it. A multicast group is a set of k hosts in a network N, which is addressable collectively by a unique group address.

Message dissemination is assumed to be genuine multicast, i.e., a source sends a message m once to the network interface in a multicast enabled backbone, which replicates m at multicast-enabled routers k times on its path to $k <= n$ receivers. This stands in contrast to most prior work on ordered multicasting assuming either unicast, where a message must be sent r times from a source to the network interface to reach $k < n$ receivers, or broadcast, where all n hosts in the network are addressed and designated receivers must filter out messages targeted at them.

Four cases of group connectivity can be observed: 1) from a single source s to a single group g, denoted as (s, g); 2) to multiple groups G, (s, G); from multiple sources S to 3) a single group, (S, g); or 4) to multiple groups, (S, G). Cases 1) and 2) have a simple solution: sequence numbers fixing the ordering relation are added to outgoing messages at the source and are delivered in that order at the destinations. Cases 3) and 4) are more difficult to implement, because sending messages from one host may be independent from other hosts, whereas reception of the same messages may be interdependent and destination groups may overlap. We are interested in totally ordered multicast from multiple sources to multiple receivers or receiver groups. We assume that hosts do not fail and network partitions do not occur. Although a very specific case, we consider overlapping groups for our protocol, because it was also a focal point in previous work on ordered multicast [87, 97, 118]. Hosts in the intersection of two overlapping multicast groups should receive a messages only once, if this message is sent to both groups.

In *total order*, two messages m_1 and m_2 are sent to a receiver set Rec in the same relative order. For example, if two sources, A and B, send messages m_1 and m_2 to receiver groups G_1 and G_2, respectively, then hosts in both groups, in particular in the intersection $G_1 \cap G_2$, should receive both messages either in the order (m_1, m_2), or (m_2, m_1). *Atomic order* demands that either all or none of the hosts in Rec receive the messages. A weaker notion of total order is causal order, based on Lamport's "happened before" relation [128].

While a causal precedence relation between two messages preserves their sending order at delivery time, messages without causal linkage may still be delivered to different hosts in different order. We assume that all logical point-to-point channels between any pair of hosts are FIFO, which prevents an earlier message by the same process to be overtaken in delivery by a later message. If not provided by the network layer, FIFO-delivery over non-FIFO channels can be implemented by having the source process add a sequence number to its messages and let destinations deliver according to such sequence numbers [18].

4.4 Taxonomy and Performance Comparison

We classify predominant ordering paradigms using reliable broadcast or multicast into two main classes, as depicted in Fig. 4.1: 1) geometry-independent protocols such as *symmetric, two-phase* or *centralized* solutions, and 2) geometry-dependent protocols such as *ring-based* and *tree-based* solutions.

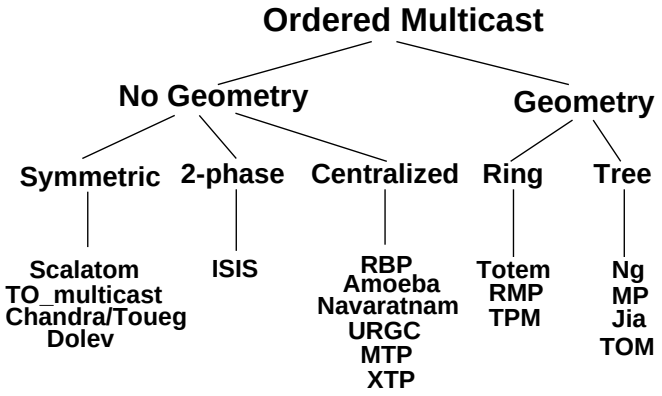

Figure 4.1: Taxonomy of ordered multicast solutions.

Some schemes may involve all hosts in the ordering process in a decentralized way, using message stability properties, versus solutions that burden one or a few hosts with the responsibility to order messages on behalf of all other hosts in a multicast group. The main problem in the first case is to reach consensus among hosts on ordering patterns, the problem in the second case is to elect sequencer nodes. Our taxonomy contrasts the distinction between symmetric and token-site algorithms proposed by Rodrigues *et al.* [186],

which also does not accommodate methods that are neither symmetric nor based on token-passing, such as tree-based ordering.

We evaluate the processing load X at involved hosts and the message overhead M required to successfully multicast a message in order from a source to all receivers. We assume IP-multicast as the dissemination model for all schemes, although all schemes except TOM have been proposed in broadcast systems. The goal of this comparison is not an elaborate modeling of the many possible nuances and optimizations of ordering schemes in conjunction with reliable multicast, but rather a simple comparison of the fundamental working structure of ordering solutions. To this end we do not include loss probabilities and assume that all schemes consistently use *sender-initiated* or *receiver-initiated* error recovery [45]. Sender-initiated models place the burden for processing acknowledgments and requests for corrupt or lost packets on the transmission source. Receiver-initiated solutions, in contrast, have retransmissions handled in local groups among receivers and sources are contacted only in cases of unrecoverable packet loss. Receiver-initiated protocols achieve better scalability because a source is likely only contacted in case of packet loss. Table 4.1 contains the notation used, where each sender is assumed to be receiver. Source nodes are denoted as SN, ordering nodes as ON, and receiver nodes as RN.

s	Number of sources sending message m to receiver set $Rec(m)$
r	Number of receivers of m in $Rec(m)$
X_f	Time to feed a packet from a higher protocol layer
X_p	Time to process the transmission of a packet (including retransmissions)
$X_\#$	Time to process a sequence number check
Y_p	Time to process a newly received packet
Y_f	Time to deliver a packet to an end process
X^ω	Processing overhead per message in protocol $\omega = \{S, 2P, C, R, T^{MP}, T^{MG}, T^{TOM}\}$
M	Number of transmissions for all receivers to receive a message orderly

Table 4.1: Analysis parameters.

4.4.1 Geometry-Independent Protocols

Reliable broadcast solutions are largely designed for fault-tolerant, asynchronous distributed systems. Such protocols are geometry-independent, i.e., all hosts are assumed to be fully

connected with each other, and routing between hosts does not presume any prearranged host geometry. We subsume symmetric, two-phase and centralized solutions under this paradigm. Centralized ordering could also be classified as a star-geometry, but the central node is typically chosen *ad hoc* based on some election or token-passing scheme among all nodes.

Symmetric Ordering

In *symmetric* schemes (S) [35, 50, 187], all hosts partake in the ordering process in a decentralized way, analogous to a voting process, using message stability properties. SN disseminate messages reliably to all hosts, which assign a timestamp to each message and place it in a pending buffer. For each message m, participant hosts (SN and RN) agree on a unique order number using timestamp information by running a consensus protocol. A message with an assigned order number is shifted to the delivery queue and delivered to end processes in the globally binding order. Thus the number of messages to be exchanged is a function of the hosts in the system involved in the ordering process. With X_c denoting the extra cost for the consensus protocol, the expected overhead of a generic symmetric protocol at SN and RN is

$$\begin{aligned} X^S_{SN} &= X_f + rX_p \\ X^S_{RN} &= s(Y_p + X_\# + rX_c + Y_f) \end{aligned} \tag{4.1}$$

With broadcast communication, a source node sends a message to $r-1$ receivers, which in turn send $r-1$ messages to agree on the final sequence number, i.e., $M_{BC} = s((r-1) + r(r-1))$, that is $O(sr^2)$ for s sources. With multicast and $r < n$ receivers, $M = s(1+2r)$, that is one multicast message to all receivers, one multicast per each of the r receivers to each other, and one timestamp sweep from all receivers to the source. Protocols with fault-tolerance measures may incur significantly higher cost [187].

Two-phase Ordering

In *2-phase* ordering (2P) [26], four communication steps are required: a source sends a message m to a multicast group, where each receiver assigns a priority number to the message, places m as pending in its local queue, and returns the priority number to the source. The source selects the highest number and sends it to all receivers, which replace the original number with the new one, tag the message as deliverable, reorder the queue and deliver messages heading the queue. The expected overhead at SN and RN is

$$\begin{aligned} X_{SN}^{2P} &= X_f + r(Y_p + X_\# + 2X_p) \\ X_{RN}^{2P} &= s(2Y_p + X_\# + X_p + Y_f) \end{aligned} \tag{4.2}$$

If we assume $r \geq s$, then $X^{2P} = max(X_{SN}^{2P}, X_{RN}^{2P}) = O(r)$. With multicast, one message from s sources to r receivers, r control messages with priority numbers back to each source, and one final control message multicast from the source to the receiver set for each message are required, i.e., $M = s(1 + r)$.

Centralized Ordering

In *centralized* ordering (C) [15, 37, 155], a source SN transmits a message m to a sequencer host, which assigns a unique number to m and forwards it to the receiver set $Rec(m)$, where it is ultimately delivered to end processes in the order prescribed by sequence numbers. The sequencer role may rotate among hosts. The expected overhead at SN, ON, and RN is

$$\begin{aligned} X_{SN}^{C} &= X_f + X_p \\ X_{ON}^{C} &= s(Y_p + X_\# + rX_p) \\ X_{RN}^{C} &= s(Y_p + Y_f) \end{aligned} \tag{4.3}$$

Hence $X^C = O(sr)$, and $M = s + r$, consisting of s messages from sources to ON, and one multicast per message from ON to all receivers. If SN = ON, we spare one step.

4.4.2 Geometry-Dependent Protocols

Geometry-dependent protocols presume a specific host topology to structure the ordering process and may also rely on the paradigms of centralized or two-phase ordering. We compare standard solutions based on rings and trees, without considering further optimizations in hybrid combinations of such geometries or other types such as hypercubes [140].

Ring-based Ordering

In *ring-based* ordering (R) [13, 179, 233], a logical ring imposes a transmission path between hosts, where each host needs only communicate with its predecessor and successor in the ring. To multicast a message, a host must possess the token; the token contains requests for messages to be resent and the highest sequence number for any message broadcast on the ring; each host maintains an input buffer containing pending messages with assigned sequence numbers; on receipt of the token, the host completes processing of messages in its buffer by adjusting sequence numbers, resends messages requested in the token, updates the token information and forwards the token; messages are sent to end processes, when marked as deliverable. Each SN, as token-site, assumes the role of ON. With X_{tk} indicating the token transfer time, the expected overhead at SN and RN in a single ring is

$$\begin{aligned} X^R_{SN} &= X_f + X_p + r(Y_p + X_\# + X_p) + X_{tk} \\ X^R_{RN} &= s(Y_p + X_\# + Y_f) \end{aligned} \tag{4.4}$$

Hence $X^R = O(r)$, if $r > s$, and the message overhead is at best $M = 2n/k$, where $2n$ is the number of token transfers required to accept k multicast messages in a ring of n nodes [179]. With $k = 1$, s sources, and despite $r < n$ receivers, $M = 2sn$.

Tree-based Ordering

For *tree-based* ordering (T), we compare the MP protocol by Garcia-Molina and Spauster [87], and the metagroup approach (MG) by Jia [118, 205] with TOM. Known tree-based reliable multicast protocols [137, 173, 237] do not feature ordering. Common to MP, MG and TOM is the idea of distributing ordering responsibility and load across several nodes on the

tree. While MP and MG use group membership information to cluster nodes for optimized message delivery, TOM uses the end-to-end multicast topology.

The MP protocol has two work phases: 1) the transmission from the source to a primary host, and 2) the transmission from this host to the receivers. It builds a forest of propagation trees, where hosts in the intersections of multicast groups are chosen as hop nodes, i.e., roots of subtrees. A message is first sent to these primary hosts, and then propagated downward in the tree toward the receiver hosts, being ordered on their propagation path, and finally unicast to the receiver hosts. The MG protocol clusters hosts from overlapping multicast groups into metagroups, which do not overlap. Each group has a primary metagroup (PM), and in each metagroup one member is assigned to be manager. Metagroups are are organized in a forest of propagation trees, such that the PM of a group is the ancestor of all other metagroups of the same group in the tree. Messages destined to multicast group G are first sent to PM(G), which propagates the messages along the tree to all other metagroups, which are subsets of G. The manager of a metagroup broadcasts a message to other members in its metagroup.

The drawback with MP and MG is the need to compute a logical propagation or metagroup tree per source as overlays to the end-to-end geometry, which means that in order to construct such a tree, the computation host(s) must know the membership of all groups. This approach works only for closed multicast and static groups, and amortizes itself only for long-lived transmissions between hosts. The processing overhead common to all tree-based schemes is

$$\begin{aligned} X_{SN}^{T} &= X_f + X_p \\ X_{ON}^{T} &= B(Y_p + X_{\#} + X_p) \\ X_{RN}^{T} &= Y_p + Y_f \end{aligned} \tag{4.5}$$

Hence generally $X^T = O(B)$, where B indicates the branching factor of the tree. With multicast, $M^{MP} = s(1+d)$ messages are required - one message from each of the s sources to the primary destination in the subtree, and one broadcast at each level of the subtree, where d is the subtree depth [87]. MG has three work phases and requires one message to

PM(G), d messages to the managers of the deepest metagroups at depth d in the subtree, and another k messages to the members of the k metagroups containing the target multicast group, i.e., $M^{MG} = s(1+d+k)$ [205]. TOM requires a multicast from s SNs to the receiver set, and p unicasts from the SN to the ON, where p is the average path length, and one final multicast from ON to RN, i.e., $M^{TOM} = s(2+p)$.

4.4.3 Results

Table 4.2 summarizes expected message costs and delays. Centralized and two-phase approaches incur only two or three message exchange phase respectively, but messaging is concentrated on specific hosts in the session, which may become a bottleneck or fail. Rings engage all hosts in a session in the transmission process, even when a source and multicast receiver group constitute only a small portion of the entire session. Trees allow for selective engagement of hosts on those subbranches or local groups, which are actually affected by the message processing.

Protocol	X	M
Symmetric (S)	$O(sr^2)$	$s(1+2r)$
Two-Phase (2P)	$O(r)$	$s(1+r)$
Centralized (C)	$O(sr)$	$s+r$
Ring-based (R)	$O(r)$	$2sn$
Tree-based (MP)	$O(B)$	$s(1+d)$
Tree-based (MG)	$O(B)$	$s(1+d+k)$
Tree-based (TOM)	$O(B)$	$s(2+p)$

Table 4.2: Average processing overhead X and multicast message cost M.

We assume that there are as many sources as receivers, $r = n$ and $s = 1$. In the graph, we neglect the cost to compute and maintain the propagation infrastructure, which may be substantial for MG and MP in comparison to TOM, which simply relies on a given acknowledgment tree. We vary the session size between $n = [1, 1000]$, with $r = n/10$ as the average size of a receiver multicast group. The MP tree depth has been projected between $d = [1, 8]$ for simulations with $n = 200$ and average group size $g = [5, 40]$ [87]. The tree depth for a metagroup tree has been projected between $d = [1, 5]$ for up to 40 metagroups

with $g = 50$, and an overlapping degree of 10. We also assume that each source sends only one multicast message per transmission cycle.

Simulations for the LORAX protocol have indicated that optimal ack trees are built when each nodes supports at least $B = 5$ neighbors [137]. As a baseline comparison, we hence choose the average depth of a subbranch in a MP and MG tree as $d = log_B r$, where $B = 5$ is the average node degree. The average path length for TOM is chosen as $p = h/2$, because roughly half of the height h of the tree needs to be traversed to converge on a ON. Note that a message comparison provides a limited view on the relative performance of the protocols, because parallelism in message processing, the processing overhead at various nodes, and the shape of the tree would need to be considered in a more precise way. However, concentrating on M alone is sufficient to express fundamental differences between the approaches. Figure 4.2 plots the multicast message cost of the various schemes under given assumptions.

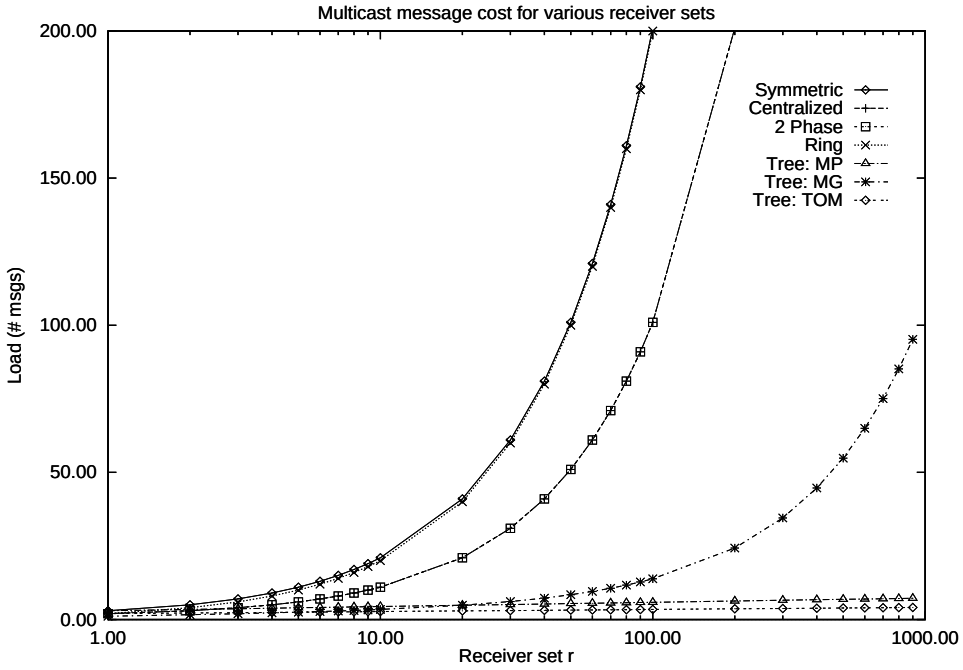

Figure 4.2: Average message cost with multicast.

The obtained results picture ordering for genuine multicast with one source transmitting. The multiple source case would reinforce that the throughput of a generic tree-based protocol for ordered reliable multicast scales better with varying receiver sets due to distributed locus and execution of sequencing. Accordingly, symmetric and ring-based methods exhibit the least scalability, because all nodes are involved in processing messages from all other nodes. However, rings may permit higher concurrency, with the drawback that latency increases in large sessions. If all nodes broadcast at the same time, latency may be low, but a consensus protocol must be run. Two-phase and centralized solutions have similar message overhead. The centralized ordering method is reasonable for few hosts, but is a potential bottleneck and single point of failure, particularly for large sessions. A logical hop between hosts in MP and MG may be multiple hops across long distances in the underlying multicast routing tree, in contrast to TOM, which operates under the assumption that the structure of the ack tree mirrors the path information in the multicast routing tree, rather than using separate propagation graphs. Comparing the three tree-based methods, TOM performs equal or better than MG and MP, spreading the computational load of ordering over multiple nodes in the tree, and is well-suited for dynamically changing multicast groups, rather than catering to static membership and long-lived transmissions.

4.5 Tree-based Ordered Multicast Protocol (TOM)

In this section, we describe a solution for the ordered multicasting problem using the idea of staggered ordering of messages on their delivery paths from sources to receivers in the reliable multicast tree, which is also used for logical connectivity between hosts for the purpose of error recovery. In contrast to earlier work, our protocol does not require construction of a separate logical propagation graph or global clock synchronization, and ordering is distributed across nodes on the delivery paths between sources and receivers in the multicast tree. For better load distribution, resilience, and ordered subcasting of messages within multicast groups, sequencer nodes are elected dynamically, based on address extensions to hosts in the multicast tree.

We assume that a reliable, unordered multicast protocol is running at every host providing reliable delivery of a message to all operational hosts in a target multicast group. Ordered multicast should be *host minimal*, i.e., no other hosts should be affected by multicast of a message than the source and receivers, and *message minimal*, i.e., the message size is a function of the size of the receiver set and not of an entire session or network [187]. Total order multicast in a broadcast model is for instance not host minimal. We subscribe to the view that ordering can be provided as middleware complementing reliable multicasting to motivate reusable coding and easier deployment, as shown in Figure 4.3. We justify our middleware approach with the observation that many networked multimedia applications are based on similar media characteristics and a generic ordered delivery semantics. Our approach contrasts the implementation philosophy taken with the MBone whiteboard tool [76], which uses Application-Level Framing (ALF) and the reliable multicast protocol SRM to provide ordering of packets within the application itself. The ALF principle states that information should be packetized into Application Data Units rather than Packet Data Units, representing a transmission unit, control unit, and processing unit at the same time.

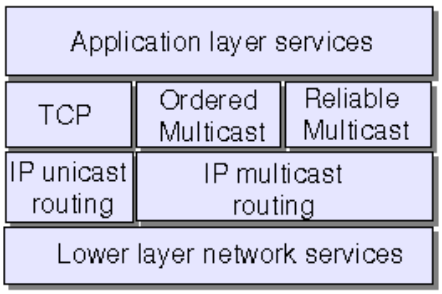

Figure 4.3: Network protocol stack with ordered multicast.

The Tree-based Ordered Multicast (TOM) protocol relies on an underlying reliable multicast tree for propagation of ordering information besides acknowledgments and retransmissions. This tree is assumed to approximate an underlying shared multicast routing tree, which for the Internet is built using various protocols such as CBT [19] or PIM-SM [73]. For the following description, we assume that hosts do not fail and network partitions do not occur. Trees can be constructed per source, which amortizes itself only for long-lived or

large-volume transmissions, or dissemination can be based on a shared tree, across which (negative) acknowledgments are relayed between hosts. In such a tree, sources may change frequently, only one collective infrastructure must be maintained, and a source need not know the identity of all receivers in the multicast group. However, the paths from sources to receivers may be suboptimal.

It is unimportant for the description of the ordering mechanism which reliable multicast protocol is used. It is also not crucial whether the end-to-end multicast tree is source-based or shared, but we will exemplify how TOM operates to provide total order in a shared tree. The key idea in TOM is to multicast a message from a source to a receiver set, combined with sending ordering information for the message (sequence numbers or time stamps) to a common node on the tree elected as ordering node for this receiver set (or multicast group). The ordering node sequences messages assigned to it, and multicasts binding sequence numbers for final delivery to the receiver set, where pending messages are to be delivered. TOM can be deployed in the form of an API accessible to applications with ordering needs.

4.5.1 Data Structures

A host in the multicast tree is either a source node (SN), an extra node (EN), a primary node (PN), an ordering node (ON), or a receiver node (RN). Since every host in the multicast session runs the ordering protocol, roles are assumed on-the-fly and no dedicated hardware is needed. SN emit messages to one or more multicast groups in a session. EN are nodes, which are not a member of the receiver set for a message, and relay messages upward or downward in the tree without participation in the ordering process. PN are hosts on the upward ordering path from SN to ON, aggregating control messages in local order and forwarding revised sequence numbers up in the tree. The ON is the sequencer node for a message, gathering sequence number bids set on route by PN, deciding on a globally valid number, and multicasting the message to the receiver set with a final and binding sequence number directive. Sources can be ON as well. RN are message recipients, delivering

them to end-processes according to an ON-sanctioned sequence number. Nodes can be SN for their own messages and assume all other roles for other messages. Edges in the acknowledgment tree point from children nodes to their parents. It is possible to associate ordering responsibilities with specific, centralized nodes. In aforementioned protocols, such nodes would be the cores in CBT, rendezvous points in PIM-SM, domain managers in TMTP, or designated receivers in RMTP, however, we strive to achieve more flexibility in allocating ordering responsibilities ad hoc and independent of the nodes involved in group membership management and error recovery.

A TOM message $m = (m^h, m^b)$ consists of a control header m^h and body m^b, with $m^h = (SN\_id, Rec, seq\#, ts, of)$, where SN_id is the source identifier, Rec is the target receiver set (which is either a multicast group, or a collection of individual node identifiers), $seq\#$ is the sequence number used for ordering, ts is an optional timestamp for ordering using timing information at nodes, and of is the ordering flag indicating that a binding sequence number for the message has been set. m^b contains the payload.

Each node maintains two message windows for ordering: a window for unordered messages (uw), which have been received, but whose delivery is pending, and an ordered messages window (ow) for messages, which are correctly ordered and can be delivered to local processes. The sizes of these buffers are limited by the number of hosts in the largest multicast group known at the time of buffer allocation. Each host programs its local network interface to subscribe to multicast packets on the same local network, or to receive packets from routers based on IGMP [74] information.

4.5.2 Operation

TOM performs message ordering in four steps: 1) a message is multicast from each SN to receivers; 2) a control message is unicast from SN across PN to the ON for the designated multicast group or transmission, where PN aggregate messages from their subtrees and hence stagger the ordering process upward in the tree; 3) a binding sequence number for this message is determined and multicast to the receiver group; and 4) messages are deliv-

ered to end hosts according to the agreed-upon sequence numbers. The goal is to deliver messages consistently in an order all hosts agree to, without requiring sources to know the constituency of the receiver set. Multicast group information is assumed to be available from a session directory service.

To allow for selective addressing of hosts and dynamic election of an ON we introduce a labeling mechanism known from multiprocessor routing and recently proposed for reliable multicast in the tree-based protocol Lorax [137] and for multicast routing [136]. Labels allow for open ordered multicast, i.e., addressing of specific nodes in the tree without the need to manifest a separate multicast group or revealing IP-addresses, and facilitate self-routing of messages to their destinations based on prefix comparison. Each node i in the acknowledgment tree is labeled with a unique label $l(i)$, which is the prefix of all children of i. The label alphabet is a set of symbols with a defined order, such as integers or letters with lexicographic order, with the alphabet cardinality corresponding to the tree branching factor B. The heuristics to select an ON is as follows: for each set of messages destined to a particular multicast group or set of hosts, elect as ON the node, whose label is the longest common prefix among all node labels in the receiver set. Each ON gathers sequence number bids set en route by PNs, deciding on a globally valid number, and multicasts the respective message to the receiver set with a final and binding sequence number directive. Figure 4.4 illustrates the mechanics of TOM.

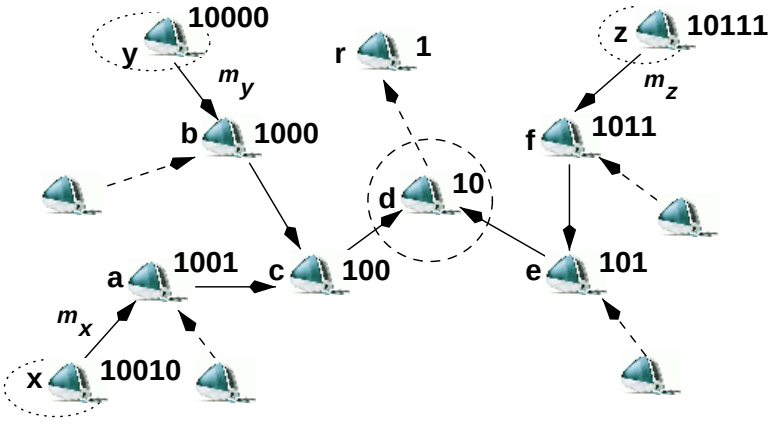

Figure 4.4: Ordered multicast on acknowledgment tree using address labels (node labels are only depicted if nodes are involved in transmission.)

Node r, as the root of the tree, carries label 1. Node d is the only child in this multicast session, carrying the prefix of its parent r concatenated with its own index 0. All three sources of messages, nodes x, y and z have labels of length 5, being positioned at depth 5 in the tree. The key idea with using labels for the ordering procedure is to create a confluence of messages at strategically optimal nodes in the tree for ordering a number of messages arriving in the same time window. Rather than depending on a statically assigned ordering node, ON is dynamically selected per transmission as the node with the longest common prefix among the sources of pending messages in the targeted multicast group, without the need to pass an election token among nodes.

Consider the case that x, y and z want to multicast messages to a multicast group $Rec = \{x, y, z, a, b, c, d, e, f\}$. Each source multicasts its message to Rec, where it is entered in the order of collective arrival into uw. Control messages m_x^h and m_y^h are routed from SN x and y, respectively, across their parents to the first common prefix node c, are intermittently ordered at c and, with revised sequence numbers, percolated up in the tree to node d, where message header m_z^h is also arriving. At any node on the path, a bitmask operation on the matching prefix indicates which messages must be up-routed or handled locally. At d it is determined that its label 10 matches the longest common prefix of SN labels $l(x), l(y)$ and $l(z)$. Hence, $ON(m_x, m_y, m_z) = d$ and node d sequences and multicasts updated message headers to Rec to signal that the associated messages can be delivered. Once each receiver in Rec receives the ordering information per message m with $of = true$ from ON, it shifts m into the ow, where the heading element is delivered to end-processes first.

Similarly, messages to a multicast group located in a left subbranch of the acknowledgment tree can be handled locally by the ON of that group, without affecting any nodes in other segments of the tree. The only overhead incurred in the ordering process is the control message unicast from SNs to some ON, plus one multicast to the receiver set. Total order is hence achieved in a diffusing computation, where the ordering process is carried out along with the message multicast, but neither are receiver nodes burdened with sorting out messages, nor do they have to know the identity of ON. Through the percolation process

from SN to ON, usage of the same sequence number for a specific message to all receivers in a multicast group is guaranteed.

Labels allow open ordered multicast, i.e., addressing of specific nodes in the tree with an ordered message sequence without the need to manifest a separate multicast group, and for self-routing of messages to their destinations based on prefix comparison. Figure 4.5 specifies the ordering algorithm of *TOM()* that an on-tree host i may use to send a message m totally ordered to a receiver set Rec (hosts are assumed to carry prefix labels). Procedure *TOM_send()* multicasts a message to the receiver set and unicasts the control header towards the dynamically elected ON; *TOM_cast()* self-routes messages to a receiver based on prefix labels; and *TOM_receive()* checks, whether a node is EN, PN, ON, or RN and takes according actions:

4.5.3 Causal and Atomic Delivery

Consider the special case of ordering with this mechanism, when messages must be sent to two different, but overlapping multicast groups, e.g., $G_1 = \{a, b, c\}$ and $G_2 = \{c, d, e, f\}$, with $G_1 \cap G_2 = c$. Nodes in each group must receive a given message sequence in total order, and node c must not receive contradictorily ordered messages. This case can be solved if individual membership in target groups is known. Instead of choosing the node with the longest common prefix as ON, the nodes with multiple membership will then be the ordering cores for a transmission, prescribing their sequencing decisions to their respective ON. In our case, c will be instrumental in informing d about the sequence in group G_1, such that d can construct a sequence from it, which is compatible with G_2.

While total order of messages within one or more destination multicast groups is ensured, causal order among messages is not preserved in the above algorithm. To provide causality, the sequence numbers of messages to be ordered must encode causal dependency information before reaching ON. This can be achieved for instance by Lamport clocks maintained by all nodes belonging to a multicast group, and updating sequence numbers in the staggered ordering process to preserve the causal relations. To implement atomicity in delivery, that

```
proc TOM (node i)
Cobegin
    TOM_send(); TOM_receive()
Coend
proc TOM_send (message m^b, receivers Rec)
Begin /* i is SN */
    If m ≠ 0 And i = SN(m)
    Then m^h = (l(i), Rec, seq_#, _, false)
        m = (m^h, m^b)
        reliable_multicast(m, Rec)
        TOM_cast(m^h, parent(i))
End
proc TOM_cast (message m, receiver rec)
Begin /* self-route from node i to rec */
    If |l(i)| > |l(receiver)|)
    OrIf l(i) ≠ prefix(rec)
    Then If ∃ parent(i)
        Then unicast(m) to parent(i)
    Else If ∃ children(i)
        Then unicast(m) to child(i)
            where l(child(i)) = prefix(l(rec))
End
proc TOM_receive (message m, receivers Rec)
Begin
    If i ∉ Rec(m) /* i is EN */
        Then unicast(m) to parent(i);
    ElseIf of(m) = false And m^b = 0 /* i is PN */
        Then tag m with new seq#;
            TOM_cast(m, parent(i))
    ElseIf l(i) ∈ Rec /* i is RN */
        Then If m^b ≠ 0 And of(m) = false
            Then insert (m^b, uw)
            Else shift m from uw to ow
            deliver(head(ow), local processes)
    Else compute longest common prefix lcp = (m, pm)
        If lcp = l(i) /* i is ON */
        And (query parent(i) for pending msgs = 0)
        Then Forall msgs in uw select seq#s
            shift msgs with set seq#s into ow
             TOM_cast(head(ow, Rec))
End
```

Figure 4.5: TOM procedures for ordered multicast from node i to other nodes and for processing received messages from other nodes for ordered local delivery.

is, either all RN in $Rec(m)$ receive message m or none, another message exchange between all RN and ON must be introduced, where all RN signal reception of m^h and m^b to ON, and ON multicasts another `ok_to_deliver(m)` signal to all RN in the group to collectively proceed with the delivery of m^b.

4.5.4 Resilience

Resilience is another important aspect in TOM operation that we only briefly discuss for space reasons. Ordering can be linked with several types of reliability [87], including 1) giving no guarantee on the reliability of ordered deliveries, 2) assuming only inconsistent deliveries with failed hosts, 3) inciting roll-backs at operational hosts to repair inconsistent deliveries, and 4) the assumption that inconsistencies never happen. Furthermore, another set of choices addresses the time it takes to deliver a message, and to which recipients the delivery guarantee extends. In the event of host or link failures, the ordering tree may be partitioned into subtrees, each of which may continue to run TOM. A vanished ON will be replaced by the next common node in the destination set according to the label semantics. In operational subgroups, the semantics of reliable delivery is preserved for all multicast operations. Failure and recovery events must be made known to all operational hosts in an ordered fashion. Partitioned subbranches of the ordering tree may rejoin as soon as communication paths between them are reestablished. A link failure is detected, when a host fails to probe a neighbor node on the tree before expiration of a local timer. A host failure is detected when a host with a pending queue of messages does not receive an expected message within a timeout period.

4.6 Discussion

In this chapter, we proposed adding ordering services to tree-based concurrent reliable multicast. Ordered delivery of multimedia data from multiple sources is essential for a growing number of Internet applications supporting near-synchronous information sharing, such as distributed interactive simulations or distance learning environments, where ordered delivery preserves data consistency and the coherency of group activities. Previous work on end-to-end multicasting protocols has circumvented the issue of ordering by attributing it to the application layer. We argued that support for ordering below the application level allows for more rapid design and deployment of shared applications.

A novel taxonomy of ordered broadcast and multicast solutions and a basic comparison of message complexities indicated that using the underlying infrastructure of trees predominant in current IP-multicasting solutions achieves the same or better efficiency in comparison with previous approaches. The TOM protocol is a blueprint for an ordering scheme relying on a mirror copy of a logical tree geometry used for concurrent, reliable multicast. The protocol consequently adapts to rapid changes in the multicast topology and surpasses contending solutions in terms of scalability, efficiency and practicality. TOM uses aggregation of ordering primitives to minimize coordination traffic among nodes, in resemblance to a two-phase ordering protocol, and assigns address extensions to hosts for self-routing of messages and dynamic distribution of the processing load. This feature implies scalability and resilience for large-scale multicasting applications. Similar to the HGCP protocol introduced in Chapter 3, TOM supports ordering of messages for anonymous and overlapping receiver groups in shared trees, and permits extensions to support causal and atomic ordered multicast.

Chapter 5
Conclusion

In this dissertation, we have investigated protocol-based network support for group coordination in telecollaborative systems, in particular for interactive, real-time exchange of multimedia information. While much research has been invested in the last decade to create collaborative multimedia environments, work on floor control and ordered multicast has been comparatively scarce and dispersed. We summarize our contributions and outline opportunities for future work.

5.1 Summary and Contributions

A comprehensive framework for group coordination in networked multimedia systems has been presented. The framework has its foundation in a formal model of group coordination and collaboration, related to hierarchical session control and role-based session participation, revolving around the notion of turn-taking in interactive groupwork. Important design issues for such an architecture have been discussed in conjunction with the various media and session types and their properties. Our coordination model does not represent a panacea for the many open problems encountered in groupware, CSCW and networked multimedia systems. Rather, we intend it to be an integrative step towards a better understanding of group collaboration, and more flexible, rich services to facilitate it. Support of sessions with many users and of wide scope encounters barriers in firewalls and security infrastructures, which must be solved before group coordination services can be widely deployed.

Several aspects regarding floor control services in networked multimedia systems have been addressed. First, we proposed a model of dynamic role-based access for continuous, ephemeral information, as we incur it in multimedia systems. This serves as a foundation for a novel taxonomy of floor control protocols, entailing user and system-driven control schemes. We tackled floor control from the user's end by elaborating on interface design, reflecting on the organizational structure of face-to-face meetings, and the stages of interaction between users with regard to floor management. Well-designed floor control provides variable granularity, consistency checks and content-synchronization of workspaces, correct and fair floor assignment, adaptability to session and media changes, and graceful degradation, when individual hosts drop out of a session.

We then discussed methods to implement the various schemes, and alternate policies for role-based floor management. A comparative performance analysis for the various floor control mechanisms in conjunction with standard service policies showed that floor control in a multicast environment performs best in a tree infrastructure logically organizing the hosts. To date, little material is available containing a more detailed description or specification of a floor control mechanism, probably because floor control will not be an urgent issue to a large community of users until stable collaboration environments at a larger scale and scope and with more alluring communication capabilities are available to the Internet public. We have presented such a first specification for two protocols, one for networks without a specific host geometry, and one operating in a logical shared control tree, whose operation is correlated to a tree-based, end-to-end reliable multicast protocol. Some of these developments and results on floor control have been tested in the collaborative visualization system CSpray, a web-based control interface CCam for collaborative usage of a motorized camera, and a prototype for a distributed shared whiteboard.

A case has been made for adding ordering services to tree-based concurrent reliable multicast, based on the notion that ordered delivery of multimedia data is essential to a growing number of Internet applications supporting telepresence and near-synchronous information sharing. Looking at reliable multicasting for such applications, we observed

that ordering services have not been considered as an integrated component in data dissemination. We proposed a taxonomy for ordering schemes integrating reliable broadcast and multicast solutions. A simple performance comparison showed that ordering in trees surpasses contending solutions in terms of scalability, efficiency and practicality.

The TOM protocol stands in contrast to previous reliable broadcast solutions tailored to local area networks, where ordering is performed assuming symmetric communication, or exchange of messages based on centralized, ring-based or propagation geometries. Collaborative applications can be developed faster by using such ordering middleware services, rather than having to implement their own ordering apparatus. TOM is a first step in the integration of middleware services with underlying transport structures, such as reliable multicast trees, using staggered ordering of messages on their paths from sources to the receivers. Workload is hence distributed, the infrastructure used for ordering is cohesive with the one for reliability provision, and the addition of address labels yields efficient ordering for multiple groups and subgroups. Opposite to previous solutions, TOM does not require computation of separate graphs for propagating ordering information. It implements ordering in a diffusing computation, where messages are ordered on their delivery paths from sources to receivers, and each node deals only with its children and parent node instead of the entire multicast group. TOM can be adapted to various tree-based multicast protocols proposed in recent years.

5.2 Future Work

The work in this thesis can be furthered in various directions.

Floor control is a promising methodology to improve cooperative behavior between users to utilize distributed system components and administer interactive groupwork more effectively. For lack of user evaluation and experience with the design of coordination protocols, our proposed methods for evaluating such mechanisms must be complemented by simulation and field tests. Future work on floor control may focus on more sophisticated protocols to direct handoff between users, concurrent assignment of floors for the same media

within non-conflicting subsets of a multicast group, fault tolerance measures in control, and improved user interfaces integrating global and localized views of shared information in both the spatial and temporal dimension, so that users can individually size their "portholes" into collaborative space, and go back in time to trace or undo collaborative actions.

Voice, video or other media types and their interplay in relation to conference conduction and floor control must be studied to better understand what type of floor control mechanism and interface are appropriate. Interaction patterns vary with each resource and turn-taking for specific media may be predictable based on observed interaction patterns. For instance, an adaptive floor control model for speech could predict the next speaker using syntactic segmentation, recognizing rhetorical pauses vs. actual conclusions of an utterance. Time zones and cultural differences are a major problem in establishing global conferencing sessions. Mechanisms could be devised to allow users to replay and edit collaborative work from other groups asynchronously and subsequently integrate such processes in synchronous work processes, interfacing offline and online collaborative work.

Fairness is important for successful implementation of floor control as well, considering that users on slow links or operating from large distances may not be as successful in obtaining the floor as other more closely located users. Fairness is largely an issue of finding the right match between session type, interaction modes, and floor policies. The presented bare-bones efficacy analysis can be extended to include queuing of requests, and more elaborate metrics to cope with host or network failures. Measurements on traffic flow and floor exchanges in current real-world telecollaboration systems must be undertaken to validate the results. An online library of API routines may be useful for rapid prototyping of CME, which developers of collaborative systems could tap into to integrate floor control support for specific tasks and session models. At the borderline to distributed artificial intelligence, the integration of adaptive floor control with goal-based collaboration, such as communicator, synchronizer, and archiver roles in workflow models [98], and with agent-based collaboration [65], using negotiation patterns to agree on floors, is another salient research area.

Our work is more geared toward the underlying network mechanisms to provide group coordination support; however, human factors and CSCW studies should complement these technical developments to ensure that systems are accepted by users. Mechanisms to intelligently filter, jointly review and pass on collaborative work efforts are another extension of this work. With regard to user interfaces, novel ways of presenting shared multimedia content and foster telepresence and mutual awareness need to be investigated. Telecollaborative interfaces, currently still relying on the windows, icons, mouse and pointer look and feel, may be strongly improved by adaptation to 3D, virtual reality, and multi-modality to achieve more realistic, natural and immersive telepresence. New forms of signaling and manifesting coordination and control in such interfaces will be needed. For instance, the integration of network coordination support with graphics applications, such as collision control in virtual reality systems [110] may achieve a more realistic representation and control of movement in virtual territories and within object boundaries.

Extension of our work on ordered multicast may address adaptation to novel methods for reliable multicasting, more sophisticated performance comparisons including packet loss probabilities, and taking into account fault tolerance, as it has been investigated in the literature for reliable broadcast protocols. Simulation studies and development of tools using ordered multicast are still lacking, as well.

The group coordination architecture described lacks security measures. Work on secure collaboration in the form of authentication and encryption is essential for synchronous multiparty, multipoint interaction in intranets, and even more so for Internet collaboration. While forging of control information or tapping into collaborative discourse may be of limited success when hosts exchange only incremental pieces of information about local shared workspaces without context, intruders may still obtain valuable pieces of information. Since there may be no universal collaborative application suitable for all types of groupwork, much work has to be invested in rationalizing and streamlining the process of lightweight collaborative software development, both for transparent and dedicated CME.

Considering the importance of temporal relationships in delivery and joint presentation of multimedia streams, spatiotemporal synchronization for highly-interactive multicast groups under the premises of scalability and adaptability towards network changes is another building block in a group coordination architecture that our work can be extended towards. In particular, previous work on synchronization has largely excluded the problems of rapidly changing multimodal sources, variable receiver groups, and ad hoc content in multicast groups in the absence of a global clock.

Telecollaboration is still in its infancy, both in understanding the process of working together in a networked multimedia system, as well as in the network and systemic technology facilitating it. Its impact on computing and the ways how people relate through computers will become more tangible in the coming years. With the surge of ubiquitous computing [231], group coordination services for mobile devices [79] will be needed to support collaboration with hand-held devices and wearable computers. In contrast to the many current monolithic protocols serving as experimental platforms for networked activity coordination, compactness, modularity and fast deployment may be important factors for future lightweight mobile collaboration support. Web-based collaboration is another young research field, which evolves hand-in-hand with developments in web-centric programming languages. Any of the proposed work extensions must go along with deepened studies on human factors and user interaction in computer-supported cooperative work systems as a foundation for understanding synchronous group collaboration. In conclusion, we project that sophisticated coordination and collaboration services in networked multimedia systems will be an important component in "killer applications" for upcoming years in computing, and deserve focused research efforts.

Bibliography

[1] H. M. Abdel-Wahab, S.-U. Guan, and J. Nievergelt. Shared workspaces for group collaboration: an experiment using Internet and UNIX interprocess communications. *IEEE Comm. Mag.*, 26(11):10–16, Nov. 1988.

[2] S. Aggrawal and S. Paul. A flexible protocol architecture for multi-party conferencing. In *Proc. 5th Int. Conf. on Computer Communications and Networking*, pages 81–91, Rockville, Maryland, Oct. 1996.

[3] D. Agrawal and A. El Abbadi. Ordered sharing: A new lock primitive for database systems. *Information Systems*, 20(5):361–92, July 1995.

[4] D. Agrawal, J. L. Bruno, A. El Abbadi, and V. Krishnaswamy. Managing concurrent activities in collaborative environments. Technical Report TRCS94-05, UCSB, Santa Barbara, CA, 1994.

[5] L. Aguilar, J. J. Garcia-Luna-Aceves, D. Moran, E. J. Craighill, and R. Brungardt. Architecture for a multimedia teleconferencing system. In *Proc. ACM SIGCOMM*, pages 126–136, Aug. 1986.

[6] S. R. Ahuja and J. R. Ensor. Coordination and control of multimedia conferencing. *IEEE Comm. Mag.*, 30(5):38–43, May 1992.

[7] S. R. Ahuja, J. R. Ensor, and S. E. Lucco. A comparison of application sharing mechanisms in real-time desktop conferencing systems. *SIGOIS Bulletin*, 11(2-3):238–48, Apr. 1990.

[8] R. Aiello, E. Pagani, and G. P. Rossi. Causal ordering in reliable group communication. In *Proc. ACM SIGCOMM*, pages 106–115, San Francisco, CA, Sept. 1993.

[9] I. F. Akyildiz and W. Yen. Multimedia group synchronization protocols for integrated services networks. *IEEE J. on Sel. Areas in Comm.*, 14(1):162–173, Jan. 1996.

[10] M. Alfano and R. Sigle. Controlling QoS in a collaborative multimedia environment. In *Proc. of the 5th Int. Symposium on High-Performance Distributed Computing (HPDC-5*, Syracuse, NY, Aug. 1996. IEEE.

[11] M. Altenhofen, J. Dittrich, R. Hammerschmidt, T. Kappner, C. Kruschel, A. Kuckes, and T. Steinig. The BERKOM multimedia collaboration service. In *Proc. ACM Multimedia'93*, pages 457–463, Anaheim, CA, Aug. 1993.

[12] E. Amir, S. McCanne, and R. Katz. Receiver-driven bandwidth adaptation for light-weight sessions. In *Proc. ACM Multimedia*, pages p.415–426, Seattle, WA, Nov. 1997.

[13] Y. Amir, L. E. Moser, P. M. Melliar-Smith, D. A. Agarwal, and P. Ciarfella. Fast message ordering and membership using a logical token-passing ring. In *Proc. Int. Conf. on Dist. Comp. Sys. (ICDCS)*, pages 551–560, Pittsburgh, PA, May 1993. IEEE.

[14] V. Anupam and C. Bajaj. Collaborative multimedia scientific design in Shastra. In *Proc. ACM Multimedia*, pages 447–480, Aug. 1993.

[15] S. Armstrong, A. Freier, and K. Marzullo. Multicast transport protocol. RFC 1301, Feb. 1992.

[16] L. C. Austin, J. K. Liker, and P. L. McLeod. Determinants and patterns of control over technology in a computerized meeting room. In *Proc. CSCW'90*, pages 39–52, Los Angeles, CA, Oct. 1990. ACM.

[17] R. Axelrod. *The Evolution of Cooperation.* Basic Books, Inc., New York, NY, 1984.

[18] Ö. Babaoglu and K. Marzullo. *Consistent Global States of Distributed Systems: Fundamental Concepts and Mechanisms*, chapter 4, pages 55–96. In: S. Mullender (Ed.) - Distributed Systems, Addison-Wesley, 1993.

[19] T. Ballardie, P. Francis, and J. Crowcroft. Core based trees (CBT) - an architecture for scalable inter-domain multicast routing. In *Proc. SIGCOMM'93 - Communications Architectures, Protocols and Applications*, pages 85–95, San Francisco, CA, Sept. 1993. ACM.

[20] N. S. Barghouti and G. E. Kaiser. Concurrency control in advanced database applications. *ACM Computing Surveys*, 23(3):269–317, Sep. 1991.

[21] D. E. Bell and L. J. La Padula. Secure computer system: Unified exposition and multics interpretation. Technical Report MTR-2997, Mitre Corp., Bedford, MA,, July 1975.

[22] G. Bell. Telepresence. Presentation at UC/Berkeley, Feb. 1996.

[23] R. Bentley, T. Rodden, P. Sawyer, and I. Sommerville. Architectural support for cooperative multiuser interfaces. *Computer*, pages 37–46, May 1994.

[24] P. A. Bernstein and N. Goodman. Concurrency control in database systems. *ACM Computing Surveys*, 13(2):185–221, June 1981.

[25] D. Bertsekas and R. Gallager. *Data networks.* Prentice Hall, Englewood Cliffs, N.J., 2nd edition, 1992.

[26] K. Birman, A. Schiper, and P. Stephenson. Lightweight causal and atomic group multicast. *ACM Trans. on Computer Systems*, 9(3):272–314, Aug. 1991.

[27] S. A. Bly, S. R. Harrison, and S. Irwin. Media Spaces: Bringing people together in a video, audio and computing environment. *Comm. of the ACM - Special Issue on Multimedia in the Workplace*, 36(1):28–47, Jan. 1993.

[28] N. Borenstein and N. Freed. MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for specifying and describing the format of Internet message bodies. RFC 1521, Sept. 1993.

[29] C. Bormann, J. Ott, and C. Reichert. Simple conference control protocol. IETF Internet Draft *draft-ietf-mmusic-sccp-00.txt*, June 1996.

[30] J. Boyd. Floor control policies in multi-user applications. In *INTERACT'93 and CHI'93 conference companion on Human factors in computing systems*, pages 107–108. ACM, 1993.

[31] M. Broy. Formalization of distributed, concurrent, reactive systems. In *Formal Description of Programming Concepts (E. J. Neuhold and M. Paul (Eds.))*, pages 319–361. Springer-Verlag, Berlin, 1992.

[32] K. S. Candan, V. S. Subrahmanian, and P. V. Rangan. Towards a theory of collaborative multimedia. In *Proc. 3rd IEEE Int. Conference on Multimedia Computing and Systems*, pages 279–282, Hiroshima, Japan, June 1996.

[33] W. Cellary, E. Gelenbe, and T. Morzy. *Concurrency Control in Distributed Database Systems.* Studies in Computer Science and Artificial Intelligence. North-Holland, Amsterdam, Netherlands, 1988.

[34] V. Cerf, P. T. Kirstein, and B. Randell. Network and infrastructure user requirements for transatlantic research collaboration. RFC 1210, March 1991.

[35] T. D. Chandra and S. Toueg. Unreliable failure detectors for reliable distributed systems. *Journal of the ACM*, 43(2):225–267, March 1996.

[36] E. Chang. Protocols for group coordination in participant systems. In *M.M. Taylor, F. Neel, and D.G. Bouwhuis (Eds.) - The Structure of Multimodal Dialogue*, pages 229–247. North-Holland: Elsevier Science, 1989.

[37] J. M. Chang and N. F. Maxemchuck. Reliable broadcast protocols. *ACM Transactions on Computer Systems*, 2(3):251–273, August 1984.

[38] C. Charlsson and O. Hagsand. DIVE - a platform for multi-user virtual environments. *Computers & Graphics*, 17(6):663–669, Nov.-Dec. 1993.

[39] D. R. Cheriton and D. Skeen. Understanding the limitations of causally and totally ordered communication. *Operating Systems Review*, 27(5):44–57, Dec. 1993.

[40] E. F. Churchill and D. Snowdon. Collaborative virtual environments: an introductory review of issues and systems. *Virtual Reality*, 3(1):3–15, 1998.

[41] E. J. Craighill, R. Lang, M. Fong, and K. Skinner. CECED: A system for informal multimedia collaboration. In *Proc. ACM Multimedia*, pages p.437–445, Anaheim, CA, Aug. 1993.

[42] E. J. Craighill, R. Lang, M. Fong, K. Skinner, and K. Gruenefeldt. SCOOT: An object-oriented toolkit for multimedia collaboration. In *Proc. ACM Multimedia*, pages 41–50, San Francisco, CA, Oct. 1994.

[43] T. Crowley, P. Milazzo, E. Baker, H. Forsdick, and R. Tomlinson. MMConf: An infrastructure for building shared multimedia applications. In *Proc. ACM CSCW*, pages 637–650, Los Angeles, CA, Oct. 1990.

[44] D. D. Ferrari, A. Banerjea, and H. Zhang. Network support for multimedia a discussion of the Tenet approach. *Computer Networks and ISDN Systems*, 26(10):1267–1280, July 1994.

[45] D. D. Towsley, J. Kurose, and S. Pingali. A comparison of sender-initiated and receiver-initiated reliable multicast protocols. *IEEE Journal on Sel. Areas in Comm.*, 15(3):398–406, Apr. 1997.

[46] K. S. Decker and V. R. Lesser. Designing a family of coordination algorithms. Technical Report TR 94-14, Univ. of Massachusetts, Amherst, MA, Aug. 1995.

[47] S. Deering. Host extensions for IP multicasting. RFC-1112, August 1989.

[48] G. Dermler, T. Gutekunst, B. Plattner, and E. Ostrowski. Constructing a distributed multimedia joint viewing and tele-operation service for heterogeneous workstation environments. In *Proc. IEEE 4th Workshop on Future Trends of Distributed Computing Systems*, pages 8–15, Lisbon, Portugal, Sep. 1993.

[49] C. Diot and L. Gautier. A distributed architecture for multiplayer interactive applications on the internet. *IEEE Network*, 13(4):6–15, July-Aug. 1999.

[50] D. Dolev, S. Kramer, and D. Malki. Early delivery totally ordered multicast in asynchronous environments. In *Int. Symposium on Fault-Tolerant Computing*, pages 544–553, Toulouse, France, June 1993. IEEE Comput. Soc.

[51] H.-P. Dommel and J. J. Garcia-Luna-Aceves. Design issues for floor control protocols. In *Proc. IS&T SPIE Multimedia Computing and Networking*, pages 305–16, San Jose, CA, Feb. 1995.

[52] H.-P. Dommel and J. J. Garcia-Luna-Aceves. Floor control for activity coordination in networked multimedia applications. In *Proc. IEEE APCC (Asian-Pacific Conference on Communications)*, pages 405–409, Osaka, Japan, June 1995.

[53] H.-P. Dommel and J. J. Garcia-Luna-Aceves. Floor control for networked multimedia applications. ACM SIGCOMM Workshop on Middleware, Cambridge, MA, http://www.acm.org/sigcomm/sigcomm95/workshop/, Aug. 1995.

[54] H.-P. Dommel and J. J. Garcia-Luna-Aceves. Floor control for multimedia conferencing and collaboration. *Multimedia Systems J. (ACM/Springer)*, 5(1):23–38, Jan. 1997.

[55] H.-P. Dommel and J. J. Garcia-Luna-Aceves. Network support for turn-taking in multimedia collaboration. In *Proc. IS&T SPIE Multimedia Computing and Networking*, pages 304–315, San Jose, CA, Feb. 1997.

[56] H.-P. Dommel and J. J. Garcia-Luna-Aceves. Comparison of floor control protocols for collaborative multimedia environments. In *Proc. SPIE Int. Symp. on Voice, Video and Data Comm.*, pages 307–318, Boston, MA, Nov. 1998.

[57] H.-P. Dommel and J. J. Garcia-Luna-Aceves. A novel group coordination protocol for for collaborative multimedia systems. In *Proc. IEEE Int. Conf. on Systems, Man, and Cybernetics, Vol. 2*, pages 1225–1230, San Diego, CA, Oct. 1998.

[58] H.-P. Dommel and J. J. Garcia-Luna-Aceves. Efficacy of floor control protocols in distributed multimedia collaboration. *Cluster Computing J.*, 2(1):17–33, 1999.

[59] H.-P. Dommel and J. J. Garcia-Luna-Aceves. Group coordination support for synchronous Internet collaboration. *IEEE Internet Computing Magazine, Special Issue on Multimedia and Collaborative Computing over the Internet*, pages 74–80, Mar./Apr. 1999.

[60] H.-P. Dommel and J. J. Garcia-Luna-Aceves. Multisites coordination in shared multicast trees. In *Proc. Int. Conf. on Parallel and Distributed Processing Techniques and Applications (PDPTA'99)*, Las Vegas, NV, June/July 1999.

[61] H.-P. Dommel and J. J. Garcia-Luna-Aceves. Ordered end-to-end multicast for distributed multimedia systems. In *Proc. 33rd Hawaii Int. Conf.on System Sciences*, Maui, Hawaii, Jan. 2000. Forthcoming.

[62] P. Dourish and V. Bellotti. Awareness and coordination in shared workspaces. In *Proc. ACM CSCW*, pages 107–114, Nov. 1992.

[63] P. Dourish and S. Bly. Portholes: Supporting awareness in a distributed work group. In *Proc. ACM CHI*, pages 541–7, Monterey, CA, May 1992.

[64] S. Easterbrook. *CSCW: Cooperation or Conflict?* Computer Supported Cooperative Work. Springer-Verlag, London, 1993.

[65] E. A. Edmonds, L. Candy, R. Jones, and B. Soufi. Support for collaborative design: Agents and emergence. *Comm. of the ACM*, 37(7):41–47, July 1994.

[66] C. A. Ellis and S. J. Gibbs. Concurrency control in groupware systems. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 399–407, Portland, OR, 1989. ACM Press, New York.

[67] C. A. Ellis, S. J. Gibbs, and G. L. Rein. Groupware - some issues and experiences. *Comm. of the ACM*, 34(1):38–58, Jan. 1991.

[68] E. Ellmer, G. Pernul, G. Quirchmayr, and A Min Tjoa. Access controls for cooperative environments. *SIGOIS Bulletin*, 15(2):24–27, Dec. 1994.

[69] R. Elmasri and S. B. Navathe. *Fundamentals of Database Systems, 2nd Ed.* The Benjamin/Cummings Publ. Company, Redwood City et. al., 1994.

[70] Inst. Electr. & Electron. Eng. IEEE standard for Distributed Interactive Simulation - Application protocols. Report, New York, NY, Aug. 1998.

[71] D. C. Engelbart. Authorship provisions in AUGMENT. In *Proc. Twenty-Eighth IEEE Computer Society Int. Conf.*, pages 465–72, San Francisco, CA, Feb./March 1984. IEEE.

[72] H. Eriksson. MBone: the multicast backbone. *Comm. ACM*, 37(8):54–60, Aug. 1994.

[73] D. Estrin, A. Helmy D. Farinacci, D. Thaler, S. Deering, V. Jacobson M. Handley, P. Sharma C. Liu, and L. Wei. Protocol independent multicast-sparse mode (PIM-SM). RFC2362, June 1998.

[74] W. Fenner. Internet Group Management Protocol, Version 2. RFC 2236, Nov. 1997.

[75] D. F. Ferguson, C. Nikolaou, J. Sairamesh, and Y. Yemini. Economic models for allocating resources in computer systems. In *S.H. Clearwater (Ed.) – Market-based control: a paradigm for distributed resource allocation*, pages 156–183. World Scientific, 1996.

[76] S. Floyd, V. Jacobson, S. McCanne, C.-G. Liu, and L. Zhang. A reliable multicast framework for light-weight sessions and application level-framing. In *Proc. ACM SIGCOMM*, pages 342–356, Cambridge, MA, Aug./Sep. 1995.

[77] F. Fluckiger. *Understanding Networked Multimedia.* Prentice Hall, Englewood Cliffs, NJ, 1995.

[78] J. W. Forgie. Voice conferencing in packet networks. In *Proc. Int. Conf on Communications (ICC'80)*, pages 21.3/1–4, New York, NY, June 1980. IEEE.

[79] G. H. Forman and J. Zahorjan. The challenges of mobile computing. *Computer*, 27(4):38–47, April 1994.

[80] G. W. Furnas and B. B. Bederson. Space-scale diagrams: Understanding multiscale interfaces. In *Proc. ACM CHI*, Denver, CO, May 1995.

[81] H. Gajewska, J. Kistler, M. S. Manasse, and D. D. Redell. Argo: A system for distributed collaboration. In *Proc. ACM Multimedia*, pages 433–440, San Francisco, Oct. 1994.

[82] J. J. Garcia-Luna-Aceves, E. J. Craighill, and L. Aguilar. MOSAIC - a model for computer-supported collaborative work. In *Proc. MILCOM 87*, pages 19–22, New York, NY, Oct. 1987. IEEE.

[83] J. J. Garcia-Luna-Aceves, E. J. Craighill, and R. Lang. An open-systems model for computer-supported collaboration. In *Proc. 2nd IEEE Conference on Computer Workstations*, pages 40–51, Santa Clara, CA, March 1988. IEEE.

[84] J. J. Garcia-Luna-Aceves, E. J. Craighill, and R. Lang. Floor management and control for multimedia conferencing. In *Proc. IEEE Multimedia, 2nd COMSOC Int. Multimedia Comm. Worksh.*, Ottawa, Can., Apr. 1989.

[85] H. Garcia-Molina. Elections in a distributed computing system. *IEEE Trans. On Computers*, 1(Jan.):48–59, C-31 1982.

[86] H. Garcia-Molina and D. Barbara. How to assign votes in a distributed system. *J. ACM*, 32(4):841–860, 1985.

[87] H. Garcia-Molina and A. Spauster. Ordered and reliable multicast communication. *ACM Trans. on Comp. Sys.*, 9(3):242–271, Aug. 1991.

[88] L. Gong. Enclaves: Enabling secure collaboration over the Internet. *IEEE Journal on Selected Areas in Communications*, 15(3):567–575, April 1997.

[89] J. Gray and A. Reuter. *Transaction Processing: Concepts and Techniques*. Morgan-Kaufmann, San Mateo, CA, 1993.

[90] S. Greenberg. Personalizable groupware: Accommodating individual roles and group differences. In *Proc. European Conf. on Computer Supported Cooperative Work (EC-SCW'91)*, Amsterdam, Sep. 1991.

[91] S. Greenberg and D. Marwood. Real-time groupware as a distributed system: Concurrency control and its effect on the interface. Technical Report 94/534/03, Dept. of Computer Science, Univ. of Calgary, Calgary, Alberta, Can., 1994.

[92] C. Greenhalgh and S. Benford. MASSIVE: A collaborative virtual environment for teleconferencing. *ACM Trans. on Computer-Human Interaction*, 2(3):239–61, Sept. 1995.

[93] I. Greif and S. Sarin. Data sharing in group work. In *Computer Supported Cooperative Work: A Book of Readings*, pages 477–508. Morgan-Kaufman, 1988.

[94] H. P. Grice. Logic and conversation. In *P. Cole and J. Morgan (Eds.) Syntax and Semantics 3: Speech Acts*. Academic Press, 1975.

[95] J. Grudin. Computer-supported cooperative work: History and focus. *Computer*, pages 19–26, May 1994.

[96] J. Grudin. Groupware and social dynamics: Eight challenges for developers. *Comm. ACM*, 37(1):93–105, Jan. 1994.

[97] R. Guerraoui and R. Schiper. Total order multicast to multiple groups. In *Proc. 17th Int. Conf.on Distributed Computing Systems*, pages 578–585, Baltimore, MD, May 1997. IEEE.

[98] J. A. Gulla and O. I. Lindland. Modeling cooperative work for workflow management. In *Proc. Advanced Information Systems Engineering. 6th International Conference, CAiSE '94*, pages 53–65, Utrecht, Netherlands, June 1994. Springer-Verlag, Berlin, Germany.

[99] C. Gutwin and S. Greenberg. Support for group awareness in real-time desktop conferences. In *Proc. of the Second New Zealand Computer Science Research Students' Conference*, Univ. of Waikato, Hamilton, New Zealand, Apr. 1995.

[100] H. Guyennet, J.-C. Lapayre, and M. Trehel. Distributed shared memory layer for cooperative work applications. In *Proc. 22nd Annual Conference on Local Computer Networks.*, pages 72–78, Minneapolis, MN, Nov. 1997. IEEE.

[101] M. Handley and V. Jacobson. Session Announcement Protocol. Internet Draft draft-ietf-mmusic-sap-v2-03.txt, November 1997.

[102] M. Handley and V. Jacobson. SDP: Session Description Protocol. RFC 2327, April 1998.

[103] M. Handley, H. Schulzrinne, E. Schooler, and J. Rosenberg. SIP: Session Initiation Protocol. RFC 2543, March 1999.

[104] M. Handley, I. Wakeman, and J. Crowcroft. The Conference Control Channel Protocol (CCCP): A scalable base for building conference control applications. In *Proc. ACM SIGCOMM*, pages 275–287, Cambridge, MA, Aug./Sep. 1995.

[105] B. Hayes-Roth. A blackboard architecture for control. *Artificial Intelligence*, 26(3):251–321, July 1985.

[106] S. Hayne, M. Pendergast, and S. Greenberg. Implementing gesturing with cursors in group support systems. *Journal of Management Information Systems*, 10(3):43–61, Winter 1993-94.

[107] V. Hilt and W. Geyer. A model for collaborative services in distributed learning environments. In *Proc. Interactive Distributed Multimedia Systems and Telecommunication Services (IDMS)*, pages 364–375, Darmstadt, Germany, Sept. 1997.

[108] T. Hodes, M. Newman, S. McCanne, R. H. Katz, and J. Landay. Shared remote control of a video conferencing application: Motivation, design, and implementation. In *Proc. Multimedia Computing and Networking*, pages 17–28, San Jose, CA, Jan. 1999. SPIE.

[109] A. W. Holt. Diplans: A new language for the study and implementation of coordination. *ACM Trans. on Office Information Systems*, 6(2):109–125, Apr. 1988.

[110] P. M. Hubbard. Collision detection for interactive graphics applications. *IEEE Trans. on Visualization and Computer Graphics*, 1(3):218–30, Sept. 1995.

[111] P. H. Hughes. Going off the rails: Understanding conflict in practice. In *S. Easterbrook (Ed.) - CSCW: Cooperation or Conflict?*, pages 161–169. Springer, 1993.

[112] C. Huitema. *Routing in the Internet*. Prentice Hall, Englewood Cliffs, NJ, 1995.

[113] E. A. Isaacs, T. Morris, T. K. Rodriguez, and J. C. Tang. A comparison of face-to-face and distributed presentations. In *Proc. ACM CHI*, Denver, CO, May 1995.

[114] E. A. Isaacs and J. C. Tang. Why do users like video? Studies of multimedia-supported collaboration. *Computer Supported Cooperative Work (CSCW)*, 1(3):163–196, 1993.

[115] E. A. Isaacs and J. C. Tang. What video can and cannot do for collaboration: a case study. *Multimedia Systems J.*, 2(2):63–73, Aug. 1994.

[116] H. Ishii, M. Kobayashi, and J. Grudin. Integration of inter-personal space and shared workspace: ClearBoard design and experiments. In *Proc. CSCW'92*, pages 33–42, Nov. 1992.

[117] H. Ishii and N. Miyake. Toward an open shared workspace: Computer and video fusion approach of TeamWorkStation. *Comm. of the ACM - Special Issue on Collaborative Computing*, 34(12):36–50, Dec. 1991.

[118] X. Jia. A total ordering multicast protocol using propagation trees. *IEEE Trans. Parallel and Distrib. Sys.*, 6(6):617–627, June 1995.

[119] Y.-J. Joung and S. A. Smolka. A comprehensive study of the complexity of multiparty interaction. *J. of the ACM*, 43(1):75–115, Jan. 1996.

[120] D. A. Henderson Jr. and S. K. Card. Rooms: The use of multiple virtual workspaces to reduce space contention in a window-based graphical user interface. In *D. Marca and G. Bock (Eds.) - Groupware: Software for Computer-Supported Cooperative Work*, pages 283–315. IEEE Comp. Soc. Press, 1992.

[121] U. Fritzke Jr., P. Ingels, A. Mostefaoui, and M. Raynal. Fault-tolerant total order multicast to asynchronous groups. In *Proc. 17th IEEE Sympos. on Reliable Distributed. Sys.*, pages 228–234, West Lafayette, IN, Oct. 1998. IEEE.

[122] M. F. Kaashoek, A. S. Tanenbaum, S. F. Hummel, and H. E. Bal. An efficient reliable broadcast protocol. *Operating Systems Review*, 23(4):5–19, Oct. 1989.

[123] T. Kamita, S. Ichimura, K. Okada, and Y. Matsushita. A database architecture and version control for group work. In *Proc. IEEE 27th Hawaii Int. Conf. on System Sciences*, volume III, pages 438–447, Wailea, HI, USA, Jan. 1994.

[124] N. Kausar and J. Crowcroft. End-to-end reliable multicast transport protocol requirements for collaborative multimedia systems. In *Proc. 17th IEEE Symp. on Reliable Distributed Systems*, pages 425–430, West Lafayette, IN, Oct. 1998. IEEE.

[125] L. Kleinholz and M. Ohly. Supporting cooperative medicine: the BERMED project. *IEEE Multimedia*, 1(4):44–53, Winter 1994.

[126] R. Kling. Cooperation, coordination and control in computer-supported work. *Commun. of the ACM*, 34(12):83–88, Dec. 1991.

[127] R. T. Kouzes, J. D. Myers, and W. A. Wulf. Collaboratories: doing science on the Internet. *Computer*, 29(8):40–46, Aug. 1996.

[128] L. Lamport. Time, clocks, and the ordering of events in a distributed system. *Comm. ACM*, 21(7):558–565, July 1978.

[129] L. Lamport. The mutual exclusion problem: Part II – statement and solutions. *J. ACM*, 33(2):327–348, Apr. 1986.

[130] B. Lampson. Protection. *Proc. 5th Princeton Conf. on Information Sciences and Systems*, 8(1):18–24, Jan. 1974.

[131] K. A. Lantz. An experiment in integrated multimedia conferencing. In *Computer Supported Cooperative Work: A Book of Readings*, pages 533–556. Morgan-Kaufman, 1988.

[132] J. Larrue and A. Trognon. Organization of turn-taking and mechanisms for turn-taking repairs in a chaired meeting. *J. of Pragmatics*, 19(2):177–196, Feb. 1993.

[133] J. C. Lauwers and K. A. Lantz. Collaboration awareness in support of collaboration transparency: Requirements for the next generation of shared window systems. In *Proc. SIGCHI*, pages 303–311, Seattle, WA, Apr. 1990.

[134] K.-C. Lee, W. H. Mansfield Jr., and A. P. Sheth. A framework of controlling cooperative agents. *IEEE Computer*, pages 8–16, July 1993.

[135] B. N. Levine and J. J. Garcia-Luna-Aceves. A comparison of known classes of reliable multicast protocols. In *Proc. IEEE Int. Conf. on Network Protocols*, pages 112–21, Columbus, OH, Oct. 1996.

[136] B. N. Levine and J. J. Garcia-Luna-Aceves. Improving Internet multicast with routing labels. In *Proc. IEEE Int. Conf. on Network Protocols*, pages 241–250, Atlanta, GA, Oct. 1997.

[137] B. N. Levine, D. Lavo, and J. J. Garcia-Luna-Aceves. The case for reliable concurrent multicasting using shared ack trees. In *Proc. ACM Multimedia*, pages 365–376, Boston, MA, Nov. 1996.

[138] S. C. Levinson. *Pragmatics.* Textbooks in Linguistics. Cambridge University Press, Cambridge, UK, 1983.

[139] W. Liao and V. O. Li. Synchronization of distributed multimedia systems with user interaction. *Multimedia Systems*, 6(3):196–205, May 1998.

[140] J. Liebeherr and B. S. Sethi. A scalable control topology for multicast communication. In *Proc. IEEE Infocom*, San Francisco, Mar./Apr. 1998.

[141] T. Little and A. Ghafoor. Synchronization and storage models for multimedia objects. *IEEE J. on Sel. Areas in Comm.*, 8(3):413–427, Apr. 1990.

[142] A. Lux, P. de Greef, F. Bomarius, and D. Steiner. A generic framework for human computer cooperation. In *Proc. IEEE Int. Conf. on Intelligence and Cooperative information Systems*, pages 89–97, Rotterdam, Netherlands, May 1993.

[143] M. R. Macedonia and D. P. Brutzman. MBone provides audio and video across the Internet. *Computer*, 27(4):30–36, Apr. 1994.

[144] P. S. Malm. The unOfficial Yellow Pages of CSCW - Groupware, Prototypes and Projects. www11.informatik.tu-muenchen.de/cscw/yp/, Jan. 1994.

[145] T. W. Malone and K. Crowston. The interdisciplinary study of coordination. *ACM Computing Surveys*, 26(1):87–119, March 1994.

[146] R. Malpani and L. Rowe. Floor control for large MBone seminars. In *Proc. ACM Multimedia*, pages 155–163, Seattle, WA, Nov. 1997.

[147] M. M. Mantei, R. M. Baecker, A. J. Sellen, W. A. S. Buxton, T. Milligan, and B. Wellman. Experiences in the use of a media space. In *Proc. CHI'91 - Human Factors in Computing Systems. Reaching Through Technology*, pages 203–208, New Orleans, LA, Apr./May 1991.

[148] E. Mayer. An evaluation framework for multicast ordering protocols. In *Proc. SIGCOMM, Computer Communication Review*, pages 177–187, Baltimore, MD, Oct. 1992. ACM.

[149] S. McCanne. Scalable multimedia communication with Internet multicast, lightweight sessions, and the MBone. Technical Report CSD-98-1002, Computer Science Division, U.C. Berkeley, Berkeley, CA, July 1998.

[150] S. McCanne and V. Jacobson. vic: A flexible framework for packet video. In *Proc. ACM Multimedia*, pages 511–522, San Francisco, Nov. 1995.

[151] A. McKinlay, R. Procter, O. Masting, R. Woodburn, and J. Arnott. Studies of turntaking in computer-mediated communications. *Interacting with Computers*, 6(2):151–171, June 1994.

[152] C. D. Miller and R. W. Clouse. Technology-based distance learning: present and future directions in business and education. *J. of Educational Technology Systems*, 22(3):191–204, 1993-1994.

[153] K. Minsoon, H. Sungjune, J. Sangjin, H. Sunyoung, et al. Scalable and reliable synchronous collaboration environment on CORBA using WWW. In *Proc. High-Assurance Engineering Workshop*, Washington, DC, Aug. 1997. IEEE.

[154] J. F. Nash. The bargaining problem. *Econometrica*, 28:155–62, 1950.

[155] S. Navaratnam, S. Chanson, and G. Neufeld. Reliable group communication in distributed systems. In *Proc. 8th Int. Conf. on Distributed Computing Systems*, pages 439–446, San Jose, CA, June 1988. IEEE.

[156] L. Navarro, W. Prinz, and T. Rodden. Towards open CSCW systems. In *Proc. IEEE 3rd Workshop on Future Trends of Distributed Computing Systems*, pages 4–10, Taipei, Taiwan, Apr. 1992.

[157] M. L. Neilsen and M. Mizuno. Coterie join algorithm. *IEEE Trans. on Parallel and Distributed Systems*, 3(5):582–590, Sept. 1992.

[158] J. M. Ng, H. H. S. Ip, and P. H. H. Tsang. A distributed multimedia conferencing system. In *Proc. IEEE TENCON*, pages 57–60, New York, NY, 1993.

[159] T. P. Ng. Ordered broadcasts for large applications. In *Proc. IEEE 10th Symp. Reliable Dist. Sys.*, pages 188–197, Pisa, Italy, Sep. 1991.

[160] D. G. Novick and J. Walpole. Enhancing the efficiency of multiparty interaction through computer mediation. *Interacting with computers*, 2(2):227–246, Aug. 1990.

[161] J. F. Nunamaker. Collaborative computing: The next millennium. *Computer*, 32(9):66–71, Sept. 1999.

[162] K. Obraczka. Multicast transport protocols: a survey and taxonomy. *IEEE Communications Magazine*, 36(1):94–102, Jan. 1998.

[163] B. O'Conaill, S. Whittaker, and S. Wilbur. Conversation over video conferences: An evaluation of the spoken aspects of video-mediated communication. *Human-Computer Interaction*, 8(4):389–428, 1993.

[164] J. Oikarinen and D. Reed. Internet Relay Chat protocol. RFC 1459, May 1993.

[165] M. H. Olson and S. A. Bly. The Portland experience: a report on a distributed research group. *Int. J. of Man-Machine Studies*, 34(2):211–228, Feb.. 1991.

[166] J. Ott, D. Kutscher, and C. Bormann. Capability description for group cooperation. Internet Draft draft-ott-mmusic-cap-00.txt, June 1999.

[167] J. Ott, C. Perkins, and D. Kutscher. Requirements for local conference control. Internet Draft draft-ott-mmusic-mbus-req-00.txt, June 1999.

[168] H.-G. Pagendarm and B. Walter. A prototype of a cooperative visualization workplace for the aerodynamicist. In *Proc. Eurographics*, volume 12, No. 3, pages 485–508, 1993.

[169] A. Pang, C. Wittenbrink, and T. Goodman. CSpray: A collaborative scientific visualization application. In *Proc. IS&T SPIE Multimedia Computing and Networking*, pages 317–326, San Jose, CA, Feb. 1995.

[170] F. Panzieri and M. Roccetti. Synchronization support and group-membership services for reliable distributed multimedia applications. *Multimedia Systems*, 5(1):1–22, Jan. 1997.

[171] V. L. Patel, D. R. Kaufman, V. G. Allen, E. H. Shortliffe, J. J. Cimino, and R. A. Greenes. Toward a framework for computer-mediated collaborative design in medical informatics. *Methods of Information in Medicine*, 38(3):158–176, Sep. 1999.

[172] J. F. Patterson and W. S. Meeks. Rendezvous: An architecture for synchronous multiuser applications. In *Proc. ACM CSCW*, pages 317–327. ACM Press, New York, Aug. 1990.

[173] S. Paul, K. K. Sabnani, J. C.-H. Lin, and S. Bhattacharyya. Reliable multicast transport protocol (RMTP). *IEEE J. on Sel. Areas in Comm.*, 15(3):407–421, April 1997.

[174] M. O. Pendergast. Multicast channels for collaborative applications: Design and performance evaluation. *Computer Communication Review*, 23(2):25–38, April 1993.

[175] A. Poggio, J. J. Garcia-Luna-Aceves, E. J. Craighill, D. Moran, L. Aguilar, D. Worthington, and J. Hight. CCWS: A computer-based multimedia information system. *IEEE Computer*, 18(10):92–103, Oct. 1985.

[176] A. Prakash and M. J. Knister. A framework for undoing actions in collaborative systems. *ACM Trans. Computer-Human Interaction*, 1(4):295–330, Dec. 1994.

[177] M. O. Rabin. The choice coordination problem. *Acta Informatica*, 17:121–34, 1982.

[178] B. Rajagopalan. Consensus and control in wide-area group communication. Technical report, AT&T Bell Laboratories, Holmdel, NJ 07733-3030, Oct. 1993.

[179] B. Rajagopalan and P. K. McKinley. A token-based protocol for reliable, ordered multicast communication. In *Proc. of the 8th Symposium on Reliable Distributed Systems*, pages 84–93, Seattle, WA, Oct. 1989.

[180] S. Rajan, P. V. Rangan, and H. M. Vin. A formal basis for structured multimedia collaboration. In *Proc. 2nd IEEE Conf. on Multimedia Computing and Systems*, Wash., D.C., May 1995.

[181] P. V. Rangan and H. M. Vin. Multimedia collaboration as a universal paradigm for collaboration. In *Multimedia - Principles, Systems and Applications*, pages 3–15. Springer-Verlag, Apr. 1991.

[182] K. Raymond. A tree-based algorithm for distributed mutual exclusion. *ACM Trans. on Comp. Sys.*, 7(1):61–77, Feb. 1989.

[183] W. Reinhard, J. Schweitzer, and G. Völksen. CSCW tools: Concepts and architectures. *Computer*, pages 28–36, May 1994.

[184] E. Rennison. Personalized galaxies of information. In *Proc. CHI'95*, Denver, CO, May 1995. ACM SIGCHI.

[185] H. M. Robert. *Robert's rules of order*. Bantam Books, Toronto; New York, 1986.

[186] L. Rodrigues, H. Fonseca, and P. Verissimo. Totally ordered multicast in large-scale systems. In *Proc. of the 16th Int. Conf. on Distributed Computing Systems*, pages 503–510, Hong Kong, May 1996. IEEE.

[187] L. Rodrigues, R. R. Guerraoui, and A. Schiper. Scalable atomic multicast. In *Proc. 7th Int. Conf. on Computer Comm. and Networks*, pages 840–847, Lafayette, LA, Oct. 1998. IEEE.

[188] M. Roseman and S. Greenberg. Groupkit: A groupware toolkit for building real-time conferencing applications. In *Proc. of the ACM 1992 Conf. on Computer-Supported Cooperative Work*, pages 43–50, Nov. 1992.

[189] E. C. Rosen, T. R. Haining, D. D. E. Long, P. E. Mantey, and C. M. Wittenbrink. REINAS: a real-time system for managing environmental data. *Int. Journal of Software Engineering and Knowledge Engineering*, 8(1):35–53, March 1998.

[190] J. S. Rosenschein and G. Zlotkin. *Rules of Encounter - Designing Conventions for Automated Negotiation among Computers.* The MIT Press, Cambridge, MA, 1994.

[191] H. Sacks, E. A. Schlegloff, and G. Jefferson. A simplest systematics for the organization of turn-taking for conversations. *Language*, 50(4):696–735, 1974.

[192] J. H. Saltzer, D. P. Reed, and D. D. Clark. End-to-end arguments in system design. *ACM Trans. on Comp. Sys.*, 2(4):277–288, Nov. 1984.

[193] R. S. Sandhu and E. J. Coyne. Role-based access control models. *Computer*, 29(2):38–47, Feb. 1996.

[194] V. Saraswat, J. Malcom, and C. Apple. The Presence Protocol. Internet Draft, Aug. 1999.

[195] K. Schmidt and C. Simone. Coordination mechanisms: Towards a conceptual foundation of CSCW systems design. *Computer-Supported Cooperative Work*, 5(2-3):155–200, 1996.

[196] E. M. Schooler. The impact of scaling on a multimedia connection architecture. *Multimedia Systems*, 1:2–9, 1993.

[197] E. M. Schooler. Conferencing and collaborative computing. *Multimedia Systems J.*, 4(5):210–225, Oct. 1996.

[198] E. M. Schooler and S. L. Casner. An architecture for multimedia connection management. In *IEEE 4th COMSOC Int. Workshop on Multimedia Comm.*, pages 271–274, Apr. 1992.

[199] I. Schubert, D. Sisalem, and H. Schulzrinne. A floor control application for lightweight multicast conferences. In *Proceedings of ICT'98 - International Conference on Telecommunications*, pages 130–134, Thessaloniki, Greece, June 1998.

[200] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A transport protocol for real-time applications. Internet Engineering Task Force - Internet Draft, Mar. 1995.

[201] A. J. Sellen. Remote conversations: The effects of mediating talk with technology. *Human-Computer Interaction*, 10(4):401–444, 1995.

[202] H. Shen and P. Dewan. Access control for collaborative environments. In *Proc. ACM CSCW*, pages 51–58, Nov. 1992.

[203] S. Shenker, A. Weinrib, and E. Schooler. Managing shared ephemeral teleconferencing state: Policy and mechanism. Internet Draft, Mar. 1995.

[204] S. J. Shenker. Making greed work: A game-theoretic analysis of switch service disciplines. *IEEE Trans. on Networking*, 3(6):819–831, Dec. 1995.

[205] S.-P. Shieh and F.-S. Ho. A comment on 'A total ordering multicast protocol using propagation trees'. *IEEE Trans. Parallel. and Distrib. Sys.*, 8(10):1084, Oct. 1997.

[206] S. Shirmohammadi, J. C. De Oliveira, and N. D. Georganas. Applet-based telecollaboration: a network-centric approach. *IEEE Multimedia J.*, 5(2):64–73, April-June 1998.

[207] S. Singh and J. F. Kurose. Electing "good" leaders. *J. of Parallel and Distributed Computing*, 21(2):184–201, May 1994.

[208] R. G. Smith. The ContractNet protocol: High-level communication and control in a distributed problem solver. *IEEE Trans. on Computers*, 29(12):1104–1113, Dec. 1980.

[209] P. K. Srimani and S. R. Das. *Distributed Mutual Exclusion Algorithms*. IEEE Comp. Society Press, Los Alamitos, CA, 1992.

[210] K. Srinivas, R. Reddy, et al. MONET: A multimedia system for conferencing and application sharing in distributed systems. Technical report, Concurrent Engineering Research Center, West Virginia University, Morgantown, WV, Feb. 1992. CERC-TN-RN-91-009.

[211] M. Stefik, G. Foster, D. G. Brobrow, K. Kahn, S. Lanning, and L. Suchman. Beyond the chalkboard: Computer support for collaboration and problem solving in meetings. *Comm. ACM*, 30(1):32–47, Jan. 1987.

[212] R. Steinmetz. Synchronization properties in multimedia systems. *IEEE J. on Sel. Areas in Comm.*, 8(3):401–411, April 1990.

[213] R. Steinmetz and K. Nahrstedt. Multimedia*: Computing, Communication, and Applications*. Prentice Hall, Upper Saddle River, NJ, 1995.

[214] W. T. Strayer, B. Dempsey, and A. Weaver. *XTP: The Xpress Transfer Protocol*. Addison Wesley, 1992.

[215] W. Su, J. Griffioen, and R. Yavatkar. Integrating concast and multicast communication models. In *Proc. Internet Routing and Quality of Service, vol. 3529*, pages 143–153, Boston, MA, Nov. 1998. SPIE.

[216] C. Szyperski and G. Ventre. A characterization of multi-party interactive multimedia applications. Technical Report TR-93-006, International Computer Science Institute, Berkeley, Feb. 1993.

[217] H. Takagi and L. Kleinrock. Output processes in contention packet broadcasting systems. *IEEE Trans. Commun.*, COM 33(11):1191–1199, 1985.

[218] G. Texier and N. Plouzeau. Automatic management of sessions in shared spaces. In *Proc. Int. Conf. on Parallel and Distributed Processing Techniques and Applications (PDPTA'99)*, pages 67–73, Las Vegas, NV, June/July 1999.

[219] R. Tijdeman. The chairman assignment problem. *Discrete Mathematics*, 32(3):323–330, Dec. 1980.

[220] I. Tou, S. Berson, G. Estrin, Y. Eterovic, and E. Wu. Prototyping synchronous group applications. *Computer*, pages 48–56, May 1994.

[221] M. Trehel. Synchronization of dialogue requests in cooperative distance learning. Personal Communication, 1999.

[222] R. H. Trigg, L. A. Suchman, and F. G. Halasz. Supporting collaboration in notecards. In *Groupware: Software for Computer-Suopported Cooperative Work*, pages 394–403. D. Marca and G. Bock (Eds.), 1986.

[223] D. Trossen and A. Katona. Conference protocol evaluation: a load model to develop conference control protocols. In *Proc. Multimedia Systems and Applications Vol. 3528*, pages 295–306, Boston, MA, Nov. 1998. SPIE - Int. Soc. Opt. Eng.

[224] International Telecommunication Union. Recommendation T.120 on data protocols for multimedia conferencing. *http://www.itu.int*, July 1996.

[225] H. M. Vin and P. V. Rangan. System support for computer mediated multimedia collaboration. In *Proc. ACM CSCW*, pages 203–209, Nov. 1992.

[226] H. M. Vin, P. V. Rangan, and S. Ramanathan. Hierarchical conferencing architectures for inter-group multimedia collaboration. In *ACM SIGOIS Bull., Proc. Org. Comp. Sys.*, pages 43–54, Atlanta, GA, Nov 1991.

[227] H. M. Vin, P. T. Zellweger, D. C. Swinehart, and P. V. Rangan. Multimedia conferencing in the Etherphone environment. *Computer*, 24(10):69–79, Oct. 1991.

[228] C. A. Waldspurger and W. E. Weihl. Lottery scheduling: Flexible proportional-share resource management. In *Proc. First USENIX Symp. on Operating Systems Design and Implementation*, pages 1–11, Monterey, CA, Nov. 1994.

[229] M. B. Walker. Smooth transitions in conversational turn-taking: Implications for theory. *J. of Psychology*, 110(1):31–37, Jan. 1982.

[230] K. Watabe, S. Sakata, K. Maeno, H. Fukuoka, and K. Marbara. Distributed multiparty desktop conferencing system: MERMAID. In *Proc. ACM CSCW*, pages 27–38, Los Angeles, CA, Oct. 1990.

[231] M. Weiser. Some computer science issues in ubiquitous computing. *Communications of the ACM*, 36(7):74–84, July 1993.

[232] P. Wellner. Interactive with paper on the DigitalDesk. *Comm. of the ACM - Special Issue on Computer Augmented Environments*, 36, No. 7:86–97, July 1993.

[233] B. Whetten, T. Montgomery, and S. Kaplan. A high performance totally ordered multicast protocol. In *Theory and Practice in Distributed Systems, LNCS 938*, pages 33–57, Berlin, Sept. 1994. Springer.

[234] T. Winograd. A language/action perspective on the design of cooperative work. *Human-Computer Interaction*, 3(1):3–30, 1987/88.

[235] K. H. Wolf, K. Froitzheim, and P. Schulthess. Multimedia application sharing in a heterogeneous environment. In *Proc. ACM Multimedia*, pages 57–64, San Francisco, CA, Nov. 1995.

[236] R. Yavatkar and J. Griffioen. Clique: a toolkit for group communication using IP multicast. In *First Int. Workshop on Services in Distributed and Networked Environments*, pages 132–138, Prague, Czech Republic, June 1994. IEEE.

[237] R. Yavatkar, J. Griffioen, and M. Sudan. A reliable dissemination protocol for interactive collaborative applications. In *Proc. ACM Multimedia*, pages 333–344, San Francisco, Nov. 1995.

[238] R. Yavatkar and K. Lakshman. Communication support for distributed collaborative applications. *Multimedia Systems J.*, 2(2):74–88, Aug. 1994.

[239] O. ZeinEldine and H. Abdel-Wahab. Multicasting in interconnected networks. In *Proc. Sympos. Comp. and Comm.*, pages 313–319, Alexandria, Egypt, June 1995. IEEE.

[240] C. Ziegler, G. Weiss, and E. Friedman. Implementation mechanisms for packet switched voice conferencing. *IEEE J. on Sel. Areas in Comm.*, 7(5):698–706, June 1989.